Question Selection for Crowd Entity Resolution

Steven Euijong Whang, Peter Lofgren, Hector Garcia-Molina

Computer Science Department, Stanford University
353 Serra Mall, Stanford, CA 94305, USA
{swhang, hector}@cs.stanford.edu

Abstract

We study the problem of enhancing Entity Resolution (ER) with the help of crowdsourcing. ER is the problem of clustering records that refer to the same real-world entity and can be an extremely difficult process for computer algorithms alone. For example, figuring out which images refer to the same person can be a hard task for computers, but an easy one for humans. We study the problem of resolving records with crowdsourcing where we ask questions to humans in order to guide ER into producing accurate results. Since human work is costly, our goal is to ask as few questions as possible. We propose a probabilistic framework for ER that can be used to estimate how much ER accuracy we obtain by asking each question and select the best question with the highest expected accuracy. Computing the expected accuracy is \#P-hard, so we propose approximation techniques for efficient computation. We evaluate our best question algorithms on real and synthetic datasets and demonstrate how we can obtain high ER accuracy while significantly reducing the number of questions asked to humans.

I. INTRODUCTION

Entity Resolution (ER) is the process of identifying and merging records judged to represent the same real-world entity. Often, humans are better than a computer at determining if records represent the same entity. For instance, it may be hard for a computer to determine that the “Canon EOS Kiss X6i” camera is equivalent to the “Canon EOS Rebel T4i” camera (one is the Japanese-market name, the other the North American name). Similarly, to check if two items are the same, one may have to examine their photographs or user evaluations, something that is easier for humans to do. With the advent of platforms for human computation [1], [2], it is now much easier to use humans in the ER process.

In this paper we focus on a hybrid ER approach, where humans are not asked to compare all pairs of records, but only the pairs that are the “most challenging” for the computer or the “most fruitful”. Asking humans to compare all records pairs is generally not feasible. For instance, if we have 1,000 records, there are 1 million pairs, and even if we pay humans 1 cent, the cost would be 10,000 dollars. Thus, we only want to pay humans where the “payoff” is significant. In this paper we will define our intuitive goal more precisely, and will present algorithms that determine what pairs of records should be compared
by the crowd. Note that we use humans during the ER process itself, to improve the accuracy of the resolved set of records. One could also use humans in an earlier training phase (where for instance, the parameters used during ER are determined), or in a later verification phase, where the output is checked. However, these are not the main focus of this paper.

One of the key challenges we face in finding what questions to ask is determining the relationship between the human answers and the “correct” resolution of the records. To illustrate, say we are resolving the three records shown in Figure 1. For simplicity, assume each record only contains the photograph shown. All three photos show the same person, but photo $a$ shows him at an earlier age, photo $b$ at an intermediate age, and photo $c$ at a later age. If a crowd worker sees all three photos (global information), he may be able to determine they all match. However, if we only present a pair, say the young-old pair $a - c$ (local information), he is unlikely to know it is the same person. For our work, we assume workers only see two records at a time (local information). Hence, answers may need to be further processed to obtain the correct resolution. For instance, if workers say that $a - b$ and $b - c$ are the same person (but $a - c$ does not match), then we can apply transitive closure to discover that all three photos match.

As we will see, some of the question-selection algorithms we present are computationally expensive. This fact is not surprising since, if there are $n$ records, there are $n^2$ questions to select from. However, since asking humans is time consuming, it may be worthwhile spending a few tens of seconds finding the right question(s) to ask. Furthermore, we will also discuss various techniques to improve the runtime of our algorithms.

In summary, our contributions are as follows.

- We define a Crowd ER framework and explain how humans can interact with the ER process (Section II).
- We propose a probabilistic framework for reasoning about questions to ask humans. With this framework, we can model how humans respond to questions, and the expected accuracy (i.e., expected gain) from asking a particular question. We show that computing the expected accuracy is an $\#P$-hard problem (Section III).
- We propose three algorithms for finding the question that maximizes the expected accuracy (Section IV):
II. Preliminaries

A. ER Model

We start by describing a “traditional” entity resolution process, where no humans are involved. There are many variations of ER, and our short description will not cover all of them. Nevertheless, we believe we will capture a large number of the ER processes used in practice.

An ER algorithm $E_R$ receives an input set of records $R$ and a pairwise similarity function $F$, and returns a set of matching pairs of records $E_R(R, F)$. Internally, we view the ER process as having two steps, as illustrated in Figure 2. (For now, please ignore the “Question Generation” and “Human” boxes on the right.) The input to the pairwise analysis phase (PWA) is a set of records $R = \{a, b, c, d, \ldots\}$, where each record provides some attributes of a real-world entity. PWA computes similarity values for some pairs of records, using a similarity function $F(a, c)$ provided by a domain expert. Typically, similarity values are between 0 and 1. (We assume that the function is symmetric, i.e., $F(a, c) = F(c, a)$.) Note that similarities may not be computed for all possible pairs of records. For instance, with blocking techniques, only pairs in the same “category” (e.g., books or DVDs) are compared. When similarities are not explicitly computed, they are assumed to be zero.

The global analysis phase (GBA) takes as input a set of record pairs, each with a similarity value, and clusters the records that are deemed to represent the same entity. For example, one strategy for GBA is (a) to ignore pairs whose similarity is below some given threshold value (again provided by a domain expert), and then (b) to perform a transitive closure on the remaining
pairs. In general, the GBA process can use any algorithm (not just transitive closure) to produce the final partition of records that represent the same entity. For example, Correlation Clustering [3] clusters records so that the number of candidate pairs within the same cluster plus the number of non-candidate pairs across clusters is maximized.

To illustrate the ER process, consider the four person records in Table I. Let us assume that for records $a$ and $b$, $F(a, b)$ is close to 1 because in this application having the same address and phone number is sufficient evidence that the records represent the same person. Similarly, say that $F(b, c)$ is close to 1 because sharing a passport and credit card are also strong evidence. We can assume that for all other pairs the similarity value is very small because there are no attribute values in common.

PWA computes the similarity values, and after GBA applies its threshold, only pairs $a - b$ and $b - c$ remain, and say that with transitive closure we add pair $a - c$. The output can be viewed as a partition of the input records, $\{\{a, b, c\}, \{d\}\}$ (where $r - t$ is in the output if and only if $r$ and $t$ are in the same partition). However, not all GBA processes will generate a partition. Thus, we use the pairs-of-records format for the final ER output (as opposed to a partition of the input records).

Since we will be using similarities in conjunction with human judgments (next section) it is important to make a few additional comments about the similarity function. In general, a $F(a, c)$ value close to 1 indicates it is more likely that $a$ and $c$ represent the same real-world entity. How close to 1 a similarity needs to be for it to be “significant” depends on the application. The threshold of GBA is used to determine what similarities are high enough. However, note that a $F(a, c)$ value close to 0 does not necessarily imply that $a$ and $c$ are not the same real-world entity. In the example of Table I, records $a$ and $c$ have few attributes in common and hence have a low similarity. Yet they refer to the same person. In other words, the similarity function only looks at local evidence that $a$ and $c$ are the same person. In our example, the evidence that $a$ and $c$ are the same person can only be found via a global analysis, in this case transitive closure.

To distinguish between what can be discovered locally versus globally, let us define two relationships between two records $a$ and $c$:

- $C(a, c)$: Record $a$ and $c$ are connected if based on purely their contents it can be determined that they refer to the same

<table>
<thead>
<tr>
<th>Record</th>
<th>Name</th>
<th>Address</th>
<th>Phone</th>
<th>Passport</th>
<th>Credit Card</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>Robert</td>
<td>123 Main</td>
<td>1234</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>Bob</td>
<td>123 Main</td>
<td>1234</td>
<td>abcd</td>
<td>777</td>
</tr>
<tr>
<td>$c$</td>
<td>Rob</td>
<td></td>
<td></td>
<td>abcd</td>
<td>777</td>
</tr>
<tr>
<td>$d$</td>
<td>Sally</td>
<td>5678</td>
<td>efgh</td>
<td>999</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE I**

A SET OF PERSON RECORDS
real-world entity. In the example of Table I, \( C(a, b) \) and \( C(b, c) \) hold, but \( C(a, c) \) and \( C(a, d) \) do not. In other words, we say that \( C = \{ a - b, b - c \} \).

- \( S(a, c) \): Records \( a \) and \( c \) refer to the same real-world entity. In our example, \( S(a, b) \), \( S(b, c) \), \( S(a, c) \) hold, but \( S(a, d) \), \( S(b, d) \), \( S(c, d) \) do not. Hence, \( S = \{ a - b, b - c, a - c \} \).

The \( F(a, c) \) value is a proxy for \( C(a, c) \) only (not for \( S(a, c) \)). The larger the value the more confident we are that \( C(a, c) \) holds. The smaller the value, the more confident we are that \( C(a, c) \) does not hold. In the following section we will discuss a more precise, probabilistic interpretation for \( F(a, c) \).

B. Human Interaction

We now discuss how to interact with humans in the PWA phase, as shown in Figure 2. (In Section III, we explain how the Question Generation component chooses which questions to ask.) In particular, we may ask humans certain \textit{questions} of the form “Does Record \( a \) represent the same real-world entity as Record \( c \)?” The human examines the contents of the records before answering YES or NO. We represent this question as \( Q(a, c) \).

Since a human only looks at local evidence in answering \( Q(a, c) \), we assume they are telling us if \( C(a, c) \) holds, not if \( S(a, c) \) holds. For instance, if we ask a human to compare records \( a \) and \( c \) of Table I, we expect the answer to be NO. If we ask a human to compare the very young and old photos of person \( H \) in Figure 1 (i.e., records \( a \) and \( c \)), we expect the answer to be again NO: Knowing that the two photos represent the same person requires global knowledge, i.e., some prior knowledge of what \( H \) looked like through the years. A human who does not know anything about \( H \) is unlikely to say the two photos represent the same person. In conclusion, transitive closure among answers is not necessary. For instance, we can get YES answers to \( Q(a, b) \) and \( Q(b, c) \) and a NO answer to \( Q(a, c) \).

Given that we can ask questions, there are two issues to address: (a) how do we select the questions to ask? and (b) what do we do with the answers?

Issue (a) is the main topic of this paper and will be addressed shortly. For issue (b), we proceed in the most natural way: If the answer to \( Q(a, c) \) was YES, we replace whatever value \( F(a, c) \) had computed by 1, to indicate we are now certain that \( C(a, c) \) holds. If the answer was NO, we replace \( F(a, c) \) by 0 to reflect the opposite. Once we modify the similarity values, we proceed to GBA, as before. The output of GBA will be our ER answer, in light of the questions we asked.

In this paper we assume humans do not make mistakes in answering questions. (We formalize this statement in the next subsection.) In practice, of course, humans may make mistakes, but there are well known techniques for dealing with errors. For example, we can ask the same question to multiple people and take the majority answer. We can also test humans before
giving them actually work, to ensure they are reliable and knowledgeable. There are numerous papers that deal with worker quality (e.g., [4], [5]), and hence we do not believe it is necessary to discuss those issues again here.

C. Evaluating ER

Traditionally, to evaluate the result of an ER algorithm, one uses a gold standard data set. For this data set, the correct answer is known in advance. Thus, one can run the ER algorithm on the data set, and compare the output to the correct answer.

In our setting, there are actually two interpretations for the term “correct answer” above:

- The correct answer we are given (for the gold standard data set) is the correct \( S() \) set. We refer to this set as \( S^*() \). That is, the final ER output, after global analysis, should include all the record pairs in \( S^*() \).
- The correct answer we are given is the correct \( C() \) set, referred to as \( C^*() \). That is, if we ask a human to compare two records (using local information only), the correct answer should be “yes” if the pair is in \( C^*() \).

Note that if we are given \( C^*() \), we can easily compute \( S^*() \): That is, to generate \( S^*() \), we input to GBA all pairs where \( C^*() \) holds. (We give these pairs a similarity of 1.) The result is \( S^*() \).

Note incidentally the subtle difference between defining \( C^*() \) as the gold standard (with \( S^*() \) derived from it) versus defining \( S^*() \) as the independent gold standard. In the latter case, we could define a gold standard that was not achievable with our given GBA. For instance, say we are using transitive closure for GBA, and say we define a gold standard where only \( S^*(a, b) \) and \( S^*(b, c) \) hold. Then we could never get a correct final answer, no matter how many questions we asked humans. Furthermore, with \( S^*() \) as the independent gold standard, the notion of “correct human answers” is ill defined. For example, say the \( S^*() \) gold standard tells us that \( S^*(a, b) \), \( S^*(b, c) \) and \( S^*(a, c) \) hold. Then, is \( Q(a, b) = \text{NO} \) an incorrect answer? Well, we cannot say without knowing the answers to \( Q(b, c) \) and \( Q(a, c) \). Hence, in this paper, we define \( S^* \) to be the GBA result on \( C^*() \).

In practice, one can have either type of gold standard \( S^* \) or \( C^* \). As we will see in our experiments, for one gold standard data set (Hotels), we have the correct \( C^*() \). (An expert compared pairs of records at a time.) For the second data set (Publications), we have a \( S^*() \) correct answer (i.e., we know after global analysis what records belong to the same entity).

To compare the final output \( O \) of an algorithm to \( S^*() \), one can compute a variety of metrics. For instance, precision \( P \) is the fraction of the pairs in \( O \) for which \( S() \) holds. Recall \( R \) is the fraction of all pairs for which \( S() \) holds and also appear in \( O \). The \( F_1 \) metric is the harmonic mean of these two values, i.e., \( \frac{2P \times R}{P + R} \). We use the \( F_1 \) metric in our experiments. We also use the same metric to guide our selection of what question to ask a human.
III. QUESTION GENERATION

The Question Generation (QG) component in Figure 2 determines which questions that, if answered by the human, would lead to an ER result with the maximum accuracy. (Depending on the ER algorithm, QG may return different questions that are best for improving the ER accuracy.) Since we do not know the gold standard, however, we can only estimate the accuracies of ER results. We first explain how QG uses a probabilistic model based on pairwise similarities to predict how humans will answer questions. We then define the problem of choosing the best question that maximizes the expected ER accuracy against possible human answers. Finally, we show that computing the expected accuracy is an \#P-hard problem.

A. Match Probability

In order to reason about what questions to ask humans, it is essential to estimate the probability that a worker’s answer to \( Q(a,c) \) will be YES or NO. For instance, if the answer to some questions is very likely to be NO, then there may not be much benefit in asking that particular question.

The similarity values \( F(a,c) \) appear to be related to probabilities. For instance, if \( F(a,c) \) is close to 1, the domain expert has determined that \( C(a,c) \) is likely to hold, and hence it seems reasonable to assume that a human is likely to answer YES to \( Q(a,c) \). However, in most cases \( F() \) is not strictly a probability. To illustrate, consider Figure 3 that shows the results of a simple experiment. We took 2,000 records from a Hotel dataset (see Section VI for details) and computed the similarity values for all pairs of records, using a similarity function that compares the names and street addresses of hotels. Among all possible pairs, we only used 2,141 pairs that had similarities at least 0.8 for asking questions. For this dataset, we also have a gold standard \( C^*() \) set, so we can check what fraction of pairs in a given similarity range are actually in the gold standard. For instance, in Figure 3, we have 113 record pairs in the similarity range 0.9–0.95. Among these pairs, 84 are answered YES by a human. Thus, it seems reasonable to say that if \( F(a,c) \) is in the range above, then the probability that \( a - c \) is in \( C^* \) is \( \frac{84}{113} \approx 0.74 \).

We argue that in general one can transform the output of a good similarity function into a probability function, based on training data, as illustrated. That is, we can draw a graph like the one in Figure 3 and then convert each similarity to its
corresponding probability in the graph. We assume that the actual data that needs to be resolved has the same characteristics as the training data, and hence the probabilities we use (mapped similarities) will be adequate for our needs. Keep in mind that the probability values will only be used to guide our question selection. The worst that can happen if our probability mapping is incorrect is that we may not choose the most effective questions to ask humans.

Setting the right ranges of similarities (for the mapping function) depends on two factors. On one hand, we would like each range of similarities to be large enough to have enough record pairs with similarities values within that range. On the other hand, we would like to have smaller ranges in order to accurately estimate the match probability for each similarity. Hence, a solution is to adjust the ranges of similarities to ensure that each range contains at least a minimum number of samples. In Figure 3, we evenly divided the record pairs with similarities at least 0.8 into 20 buckets.

To summarize, QG computes probability values for each pair of records (using the similarity function as discussed): \( P(a,c) \) gives us the probability that a human will answer \( Q(a,c) \) in the positive. The pairwise probabilities can be captured as a match probability graph \( P = (V,E) \) where \( V \) is the set of records and each \( a \rightarrow c \in E \) has a weight of \( P(a,c) \). And since humans make no mistakes, \( P(a,c) \) also gives us the probability that \( C^*(a,c) \) holds.

### B. Possible Worlds

Using these probabilities along with the ER algorithm \( E_R \), we can actually compute how likely an ER result is to match the gold standard. To illustrate, consider a scenario with three records, \( a, b \) and \( c \), where \( F(a,b) = 0.9, F(b,c) = 0.5, \) and \( F(a,c) = 0.2 \). Suppose that these similarities convert into the probabilities \( P(a,b) = 1, P(b,c) = 0.6 \) and \( P(a,c) = 0.2 \) (see Figure 4). We assume that the GBA of \( E_R \) performs a transitive closure on matching records.

Consider a possible output of pairwise matching records, say \( W_1 = \{a \rightarrow b, b \rightarrow c, a \rightarrow c\} \). How likely is it that \( W_1 \) is the \( C^* \) gold standard? The answer, \( Pr[W_1 = C^*] \) is simply the probability that \( a \rightarrow b \) is in \( C^* \) (that is, \( P(a,b) \)) times the probability that \( b \rightarrow c \) is in \( C^* \) times the probability that \( a \rightarrow c \) is in \( C^* \). Hence, \( Pr[W_1 = C^*] = 1 \times 0.6 \times 0.2 = 0.12 \). In general, the probability of a possible world \( W \) can be computed as

\[
p(P,W) = (\prod_{a \rightarrow b \in W} P(a,b))(\prod_{a \rightarrow b \in W} 1 - P(a,b))
\]

In a similar fashion we can compute the probability of the remaining \( C^* \) outputs, as shown in the top half of Figure 5. For the given scenario, there are three more outputs with non-zero probability: \( W_2 = \{a \rightarrow b, b \rightarrow c\} \) with probability 0.48,
Possible $C^*$: \[
\begin{array}{ccc}
\text{a} & \text{b} & \text{c} \\
0.12 & 0.48 & 0.08 & 0.32
\end{array}
\]

Possible $S^*$: \[
\begin{array}{cc}
\text{a} & \text{b} \\
0.68 & 0.32
\end{array}
\]

Fig. 5. Possible Worlds of Figure 4

$W_3 = \{a - b, a - c\}$ with probability 0.08, and $W_4 = \{a - b\}$ with probability 0.32. We call $W_1$, $W_2$, $W_3$, and $W_4$ the $C^*$ possible worlds.

Using each $C^*$ possible world $W$, we can compute a $S^*$ possible world $E_R(R, F_W)$ where $F_W$ is the similarity function corresponding to $W$ where $F_W(a, b) = 1$ if $a - b \in W$ and $F_W(a, b) = 0$ otherwise. The probability of each $S^*$ possible world $W'$ is the probability sum of each $C^*$ possible world $W$ where $W' = E_R(R, F_W)$. Continuing our example above, the bottom half of Figure 5 shows the $S^*$ possible worlds for $P$. The probability of the first, $W'_1 = \{a - b, b - c, a - c\}$, is the sum of the probabilities of the first three $C^*$ worlds that lead to $W'_1$ when transitive closure is applied. The probability of the second $S^*$ world, $W'_2 = \{a - b\}$ is the probability of the fourth $C^*$ world, which leads to $W'_2$. In a sense, the $S^*$ possible worlds are our predictions of the gold standard.

From the possible worlds we can now compute how good an ER output is expected to be. First, we define the function $F_1(O_1, O_2)$ to compute the $F_1$ accuracy of the set of edges $O_1$ against the set of edges $O_2$. We also define the accuracy function $Acc(O, W)$ as a shorthand for $F_1(O, E_R(R, F_W))$. Given a set of $C^*$ possible worlds $L$ of the probability graph $P$ and an ER algorithm $E_R$, we define the expected accuracy of an ER result $O$ against a random possible world $W \in L$ as

$$E[Acc(O, W)] = \sum_{W' \in L} p(P, W') \times Acc(O, W')$$

For example, suppose that the ER result $O = \{a - b\}$, and we are computing the expected accuracy of $O$ against the probability graph in Figure 4. Using the four possible worlds $W_1$, $W_2$, $W_3$, and $W_4$ in Figure 5, we can compute $E[Acc(O, W)]$ as

$$0.12 \times Acc(O, W_1) + 0.48 \times Acc(O, W_2) + 0.08 \times Acc(O, W_3) + 0.32 \times Acc(O, W_4) = 0.68 \times F_1(O, W'_1) + 0.32 \times F_1(O, W'_2)$$

$$= 0.68 \times \frac{2 \times 1 \times 1/3}{1 + 1/3} + 0.32 \times \frac{2 \times 1 \times 1}{1 + 1} = 0.66.$$  

C. Choosing the Best Question

Using the information in the $S^*$ possible worlds, we would like to derive an ER result that maximizes the expected accuracy against these possible gold standards. We define the best question problem as choosing the question $Q(a, b)$ that, when answered,
can maximize $E[Acc(O, W)]$ when the human answers YES with probability $P(a, b)$ and NO with probability $1 - P(a, b)$. (In Section V, we discuss the problem of asking more than one question at a time.)

How do we compute the expected accuracy when a human answers a question? Suppose that the human answered YES to the question $Q(a, b)$. According to Section II-B, we then set $F(a, b)$ to 1. Hence, we need to re-compute the ER result $O$ using the new similarity function. We denote $F[a, b, 1]$ to be identical to $F$ except that $F(a, b)$ is set to 1. We then compute the new ER result $O_1 = E_R(R, F[a, b, 1])$. We also update $P$ to $P[a, b, 1]$ where $P[a, b, 1]$ is identical to $P$ except that $P[a, b, 1](a, b)$ is set to 1. Suppose we define $W_1$ to be a random possible world from $P[a, b, 1]$. We then compute the expected accuracy $E[Acc(O_1, W_1)]$. Similarly, if the human answers NO, we can compute the ER result $O_2 = E_R(R, F[a, b, 0])$. We also update $P$ to $P[a, b, 0]$ and define $W_2$ to be a random possible world from $P[a, b, 0]$. We then compute the expected accuracy $E[Acc(O_2, W_2)]$. Finally, we define the random variable $A(a, b)$ where $A(a, b) = 1$ if the human answered the question $Q(a, b)$ and $A(a, b) = 0$ otherwise. Since the human answers YES with probability $P(a, b)$ and NO with probability $1 - P(a, b)$, the final expected accuracy given that $Q(a, b)$ is answered is

$$E[Acc(O, W)|A(a, b)] = P(a, b) 	imes E[Acc(O_1, W_1)] + (1 - P(a, b)) 	imes E[Acc(O_2, W_2)]$$

We thus ask the question $Q(a, b)$ that has the maximum $E[Acc(O, W)|A(a, b)]$ value.

As an illustration, suppose we are choosing the best question for the probability graph in Figure 4. We use an ER algorithm that matches two records if they have a similarity at least $t = 0.7$, and then performs a transitive closure on the matching records. Suppose we choose the question $Q(b, c)$. We first compute $Acc(O_1, W_1)$ where $O_1 = E_R(R, F[b, c, 1])$ and $W_1$ is a random possible world from $P[b, c, 1]$. Since we assume the human says YES to $Q(b, c)$, the ER result $O_1 = \{a - b, b - c, a - c\}$. In addition, since $P(b, c) = 1$, there is only one $S^*$ possible world: $\{a - b, b - c, a - c\}$. Hence, $E[Acc(O_1, W_1)] = 1$.

We now compute $E[Acc(O_2, W_2)]$ where $O_2 = E_R(R, F[b, c, 0])$ and $W_2$ is a random possible world from $P[b, c, 0]$. This time, the ER result is $O_2 = \{a - b\}$, and there are two $S^*$ possible worlds: $\{a - b, b - c, a - c\}$ with probability 0.2 and $\{a - b\}$ with probability $(1 - 0.2) = 0.8$. Hence, $E[Acc(O_2, W_2)] = 0.2 \times \frac{2 + 1 \times 1/3}{1 + 1/3} + 0.8 \times 1 = 0.9$. We can now compute $E[Acc(O, W)|A(b, c)] = 0.6 \times 1 + 0.4 \times 0.9 = 0.96$. In a similar fashion, we can compute the expected accuracy for $Q(a, b)$ as 0.66 and $Q(a, c)$ as 0.76. Hence, the question $Q(b, c)$ results in the highest expected accuracy. Intuitively, by asking $Q(b, c)$ where the human has a high chance of answering YES, we are likely to have the gold standard $\{a - b, b - c, a - c\}$. At the same time, the ER algorithm is also likely to produce $\{a - b, b - c, a - c\}$, resulting in high accuracy.

**Proposition 3.1.** Suppose that ER performs a transitive closure on matching records. Given a match probability graph $P$ and
an ER result $O$, computing the expected accuracy $E[\text{Acc}(O, W)]$ is $\#P$-hard (and thus $NP$-hard).

Proof: Suppose that GBA performs a transitive closure of matching records. It is known that given a graph $P = (V, E)$ and a pairwise probability $p_e \in [0, 1]$ for each edge $e \in E$, it is $\#P$-hard to compute the exact probability that the PWA possible world is connected [6]. Hence it is $\#P$-hard to compute the probability that the GBA possible world clusters all records. We will show that if we could compute the expected $F_1$ of any set of pairwise matching edges in polynomial time, then we could compute the probability that the GBA possible world is the set of all edges and conclude that $P = NP$.

Let the graph $P = (V, E)$ and a pairwise probability $p_e \in [0, 1]$ for each edge $e \in E$ be given. (Notice that if $e = a - b$, then $p_e = P(a, b).$) For each edge $e$, define a random variable $X_e$ which indicates that $e$ is in the transitive closure of the $C^*$ gold standard, i.e. $X_e = 1$ if $e$ is in the transitive closure of the gold standard and $X_e = 0$ otherwise. From these random variables, we can write down random variables for recall, precision, and $F_1$ of the ER result $O$. Precision is the fraction of edges in $O$ which are in the gold standard

$$\frac{\sum_{s \in O} X_s}{|O|}.$$  

Recall is the fraction of gold standard edges that are in $O$

$$\frac{\sum_{s \in O} X_s}{\sum_{e \in E} X_e}.$$  

The $F_1$ score is the harmonic mean of these two values:

$$F_1(O) = 2 \frac{\sum_{e \in O} X_e}{|O| + \sum_{e \in E} X_e}.$$  

Let $Y = \sum_{e \in E} X_e$ be the number of edges in the transitive closure of $C^*$. The probability that $C^*$ is connected is exactly the probability that $Y = \binom{n}{2}$, so if we could compute $\text{Pr}(Y = \binom{n}{2})$ then we could compute the probability that $C^*$ is connected, which is $\#P$-hard to compute. For each $k \leq \binom{n}{2}$ define

$$y_k = \text{Pr}(Y = k).$$  

In order to learn several different linear relations among the values $y_k$, we define a sequence of graphs with a few trivial edges added. For an integer $i \geq \binom{n}{2}$, define $P_i$ to be the result of starting with $P$ and adding $i - \binom{n}{2}$ artificial new pairs of vertices connected by a new edge. We let $p_e = 0$ for the new edges, and we let $O_i$ be all $\binom{n}{2}$ pairs of original vertices as well as the new artificial edges, so $|O_i| = i$. Then using the above expression for $F_1$ and the fact that $E \subseteq O_i$ we see that for the edge set $O_i$

$$E[F_1(O_i)] = 2E \left[ \frac{\sum_{e \in E} X_e}{|O| + \sum_{e \in E} X_e} \right].$$
Substituting, we see that

$$E[F_1(O_i)] = 2E\left[\frac{Y}{i+Y}\right] = 2 \sum_{k=0}^{(n)} \frac{k}{i+k} y_k.$$ 

Thus by choosing different values of $i$, we can obtain linear relations among the unknown variables $y_k$. In particular, if we choose a prime number $p > 2^{\frac{n}{2}}$ and let $i = p-1, p-2, \ldots, p-\frac{n}{2}$, the coefficients of the linear relations we obtain form a matrix with rational entries such that all the denominators are relatively prime to $p$ except those on the main diagonal. For example, if $n = 3$ and $p = 7$ we have the equations

$$\begin{pmatrix}
E[F_1(O_6)] \\
E[F_1(O_5)] \\
E[F_1(O_4)]
\end{pmatrix} = 2
\begin{pmatrix}
\frac{1}{7} & \frac{2}{7} & \frac{3}{7} \\
\frac{1}{7} & \frac{2}{7} & \frac{3}{7} \\
\frac{1}{7} & \frac{2}{7} & \frac{3}{7}
\end{pmatrix}
\begin{pmatrix}
y_1 \\
y_2 \\
y_3
\end{pmatrix}$$

If we multiply each row by the product of its denominators, we get an integer matrix, and if we look at this matrix modulo $p$ we get a matrix which is non-zero exactly on the main diagonal and hence invertible modulo $p$ and hence invertible. (This invertible modulo $p$ technique was suggested by Rishi Gupta.) Hence we can invert this matrix in polynomial time and solve for $y_{\binom{n}{2}}$, the value that is $\#P$-hard to compute. Thus it is $\#P$-hard to compute expected $F_1$ for arbitrary graphs and sets of edges.

Consequently, computing $E[Acc(O,W)]$ is $\#P$-hard in general. The proposition also suggests that it is $\text{NP}$-hard to compute the optimal next question, since even evaluating how useful it is to choose a particular question is $\#P$-hard.

IV. CROWD ER ALGORITHMS

We first propose an exhaustive “brute-force” algorithm for deriving the best question that has the highest expected accuracy. Since this algorithm has an exponential runtime, we propose an approximation algorithm (called GCER) that runs in polynomial time. Finally, we describe a simple heuristic for choosing questions without any information about the ER algorithm.

A. Exhaustive Algorithm

The exhaustive algorithm (Algorithm 1) iterates all the edges in $P$ and computes the expected $F_1$ for asking that question. Suppose we use the example in Section III-C. There are three possible questions to ask: $Q(a,b)$, $Q(b,c)$, and $Q(a,c)$. If we first choose $Q(b,c)$ in Step 3, we set $p = 0.6$. We first assume that the human says YES to the question $Q(b,c)$ and set $F(b,c)$ and $P(b,c)$ to 1. We then use Algorithm 2 to compute the expected accuracy of $Q(b,c)$. In Step 1 of Algorithm 2, $O = \{a-b, b-c, a-c\}$. Since the human said YES to $Q(a,b)$ and $Q(b,c)$, there are now two $C^*$ possible worlds of $P$: $\{a-b, b-c\}$ and $\{a-b, b-c, a-c\}$ with probabilities $1-0.2 = 0.8$ and $0.2$, respectively. As a result, the only $S^*$ possible world is $\{a-b, b-c, a-c\}$. Hence, after Steps 4–5, $A$ is set to $0.8 \times \frac{2 \times 1 \times 1}{1+1} + 0.2 \times \frac{2 \times 1 \times 1}{1+1} = 1$ and returned by Algorithm 2.
Algorithm 1: Exhaustive Algorithm

input: A set of records $R$, an ER algorithm $E_R$, a similarity function $F$, and a match probability function $P$

output: The question $Q(a, b)$ with the highest expected accuracy

1. $Z \leftarrow \text{null}$;
2. $M \leftarrow 0$;
3. for $a - b \in \{r - s | r - s \in R \times R \land r \neq s\}$ do
   4. $p \leftarrow P(a, b)$;
   5. $t \leftarrow F(a, b)$;
   6. $F(a, b) \leftarrow P(a, b) - 1$;
   7. $Y \leftarrow \text{ComputeAccuracy}(R, E_R, F, P)$;
   8. $F(a, b) \leftarrow P(a, b) - 0$;
   9. $N \leftarrow \text{ComputeAccuracy}(R, E_R, F, P)$;
10. $P(a, b) \leftarrow p$;
11. $F(a, b) \leftarrow t$;
12. $A \leftarrow p \times Y + (1 - p) \times N$;
13. if $A > M$ then
   14. $Z \leftarrow Q(a, b)$;
   15. $M \leftarrow A$;
16. return $Z$;

We then move on to Steps 8–9 in Algorithm 1 where we compute the expected accuracy of $Q(b, c)$ when the human answers NO. In Step 1 of Algorithm 2, $O = \{a - b\}$. In Step 3, there are two $C^*$ possible worlds of $P$: $\{a - b\}$ and $\{a - b, a - c\}$ with probabilities $1 - 0.2 = 0.8$ and $0.2$, respectively. As a result, there are two $S^*$ possible worlds: $\{a - b\}$ with probability $0.8$ and $\{a - b, b - c, a - c\}$ with probability $0.2$. Hence, after Steps 4–5, $A$ can be computed as $0.8 \times \frac{2 \times 1 \times 1}{1 + 1} + 0.2 \times \frac{2 \times 1 \times 1 / 3}{1 + 1 / 3} = 0.9$. In Step 12 of Algorithm 1, we can now compute the expected accuracy of $Q(a, b)$ as $0.6 \times 1 + 0.4 \times 0.9 = 0.96$. In Steps 14–15, we thus set $Z = Q(a, b)$ and $M = 0.96$. Next, we repeat Steps 3–15 in Algorithm 1 for the remaining edge $a - c$. It turns out that $Q(a, c)$ and $Q(a, b)$ have expected accuracies of $0.76$ and $0.66$, respectively, so we return $Q(b, c)$ in Step 16.

We now show the complexity of Algorithm 1.

**Proposition 4.1.** The complexity of Algorithm 1 is $O(|R|^2 \times |L| \times (C(E_R) + |R|^2))$ where $E_R$ is the ER algorithm, $L$ is the $C^*$ possible worlds of $P$, and $C(E_R)$ is the time complexity of $E_R$.

**Proof:** We iterate through all $O(|R|^2)$ edges (Step 3, Algorithm 1) and, for each possible world in $L$ (Step 5, Algorithm 2), we compute ER once ($C(E_R)$) and then compute the $F_1$ accuracy ($O(|R|^2)$). If all ER results are stored as partitions of records
Algorithm 2: ComputeAccuracy

Input: a set of records $R$, an ER algorithm $E_R$, a similarity function $F$, and a match probability function $P$

Output: The expected accuracy

1. $O \leftarrow E_R(R, F)$;
2. $A \leftarrow 0$;
   
   // Compute $E[Acc(O, W)]$
3. $L \leftarrow C^*$ possible worlds of $P$;
4. for $W' \in L$ do
5.   $A \leftarrow A + p(P, W') \times F_1(O, E_R(R, F_{W'}))$
6. return $A$;

Instead of sets of matching pairs, we can compute the $F_1$ accuracy in $O(|R|)$ time by meeting two partitions using hash functions.

Since $L$ is exponential in the number of edges in $P$ that have probabilities strictly between 0 and 1, the algorithm complexity is exponential to the number of records $|R|$ as well.

B. GCER Algorithm

While the exhaustive algorithm correctly returns the question with the highest expected accuracy, there is a great deal of redundant computation that makes the entire process inefficient. In this section, we propose the GCER algorithm, which improves Algorithm 1 with a number of optimizations to produce an approximate result within polynomial time.

1) Pruning Questions: We can reduce the number of questions to compare by avoiding record pairs with very high or low probabilities. For questions with probabilities less than a threshold $l$ (more than $h$), we can skip its entire loop (Steps 3–15) in Algorithm 1 assuming the human will say NO (YES). If most record pairs will not match with each other, we can significantly reduce the number of questions to consider without missing obviously good questions. The pruning optimization does not reduce the complexity of Algorithm 1, but can significantly reduce the actual runtime of GCER. On the other hand, we may now miss the opportunity to choose the question with the highest expected accuracy. In Section VI-B4, we study the tradeoffs of pruning questions.

2) Monte-Carlo Approximation for Accuracy: Since computing the expected accuracy of a question is $\#P$-hard, we now show an approximation technique for computing $E[Acc(P, O)]$ for each question and answer pair using Monte-Carlo methods. Instead of generating all possible worlds from $P$, we select a random set of possible worlds where the probability for choosing each world $W$ is the product of the probabilities of $W$’s edges in $P$. We then compute the ER result $S = E_R(R, F_W)$. Finally, we compute the accuracy $F_1(O, S)$. By repeating the sampling and ER evaluation steps, we can eventually produce an average $F_1$ accuracy that is very close to $E[Acc(P, O)]$. 
We can compute the number of samples needed to estimate the final matching probabilities of records. Let $X$ be the $F_1$ value we are estimating, so $X$ is a random variable in $[0, 1]$. We would like to know $\mu = E[X]$. We estimate the $F_1$ value by taking the mean $\bar{X}$ of the $F_1$ values from the sample ER results. Given an error tolerance of $\epsilon$ and a probability $\delta$, suppose we want $Pr(|\mu - \bar{X}| < \epsilon) > 1 - \delta$. By Hoeffding’s inequality [7], we need at least $n = \lceil \frac{\ln(2/\delta)}{2\epsilon^2} \rceil$ samples.

To illustrate the computation of $n$, suppose we want to compute $E[Acc(P,O)]$ where there are $|R|$ records. To compute $E[Acc(P,O)]$ exactly, we need to run ER $2^{|R|\times(|R|-1)/2}$ times assuming all probabilities in $P$ are strictly between 0 and 1. For the approximation, suppose we allow $\epsilon = 0.05$ and $\delta = 0.1$ (i.e., we use a 90% confidence interval). Then we need to run ER $n = \lceil \frac{\ln(2/\delta)}{2\epsilon^2} \rceil = 600$ times. In Section VI-B2, we show that a sample size of $n = 20$ is sufficient for producing highly-accurate ER results quickly. The reason is that we only need to find the question with the relatively highest expected accuracy instead of computing the exact values of expected accuracies.

By using the Monte-Carlo method, we can reduce the complexity of Algorithm 1 to be polynomial.

Proposition 4.2. Using the Monte-Carlo method for estimating the ER accuracy, the complexity of Algorithm 1 is $O(|R|^2 \times (C(E_R) + |R|^2))$.

Proof: Again, we iterate $O(|R|^2)$ edges (Step 3, Algorithm 1). For each edge $a - b$, we run ER and compute the $F_1$ accuracy $n$ times in order to compute the expected accuracy of answering $Q(a,b)$. Notice that the complexity is now polynomial to the number of records $|R|$ assuming that $C(E_R)$ is polynomial.

3) Sharing ER Results: Even with the Monte-Carlo methods in Section IV-B2, the number of questions to ask still increases in proportion to the number of edges in $P$. We can further reduce the expected number of samples to a constant by sharing ER samples among different question and answer pairs. We only compute the expected accuracy for questions with probabilities strictly between 0 and 1 (i.e., $l > 0$ and $h < 1$). Suppose that we want at least $n = 2$ samples for each question and answer pair in Figure 4. If we set $l = 0.1$ and $h = 0.9$, the possible question and answer pairs are: $[Q(b,c), YES]$, $[Q(b,c), NO]$, $[Q(a,c), YES]$, and $[Q(a,c), NO]$. Suppose that we choose a random possible world $W_1 = \{a - b, b - c\}$ from $P$. Notice that $W_1$ could also be a possible random world from $P$ when $Q(b,c)$ is answered YES and $Q(a,c)$ is answered NO. Hence, the ER result $S_1 = E_R(R, F_{W_1})$ is a possible ER sample for the two question-answer pairs above.

Using this idea, we can pre-generate random $C^*$ possible worlds $W_1, \ldots, W_m$ of $P$ and using a subset of them for computing $E[Acc(P,O)]$ for each question. For each question and answer pair $[Q(a,b), ans]$, we can choose any subset of possible worlds $W'_1, \ldots, W'_n$ where each world $W'_i$ contains (does not contain) the edge $a - b$ if $ans$ is YES (NO). Since we need at least $n$ ER samples per question and answer pair, we generate the possible worlds from $P$ until each question and answer pair has enough
samples. Continuing our example above, suppose that we now generate the possible worlds $W_2 = \{a-b, b-c, a-c\}$ and $W_3 = \{a-b, a-c\}$. We then generate the ER results $S_2 = E_R(R, F_{W_2})$ and $S_3 = E_R(R, F_{W_3})$. This time, $S_2$ is a possible sample for the question-answer pairs $[Q(b,c), \text{YES}]$ and $[Q(a,c), \text{YES}]$ while $S_3$ is a possible sample for the pairs $[Q(b,c), \text{NO}]$ and $[Q(a,c), \text{YES}]$. As a result, every question-answer pair now has at least $n = 2$ samples. Since we only consider question-answer pairs with probabilities strictly between 0 and 1, generating the ER samples is guaranteed to terminate.

**Proposition 4.3.** Given a low threshold $l$ and a high threshold $h$ for probabilities ($0 < l < h < 1$), the expected number of ER samples to create to have at least $n$ samples for each question-answer pair with probabilities between $l$ and $h$ is $\frac{n}{\min\{l,1-h\}}$.

**Proof:** Suppose that there is a question with the lowest possible probability of $l$. Then we need to generate at least $\max\{\frac{n}{1-l}, \frac{n}{h}\}$ samples. Now suppose there is a question with the highest possible probability $h$. This time, we need to generate at least $\max\{\frac{n}{1-h}, \frac{n}{l}\}$ samples. Any question with a probability larger than $l$ and smaller than $h$ requires fewer samples than $\max\{\frac{n}{1-l}, \frac{n}{1-h}\}$. Since $l \leq h$, we know that $\frac{n}{h} \leq \frac{n}{l}$ and $\frac{n}{1-h} \leq \frac{n}{1-l}$. Hence, we need to create $\max\{\frac{n}{1-l}, \frac{n}{1-h}\} = \frac{n}{\min\{l,1-h\}}$ ER samples to have an expected number of at least $n$ ER samples for each question and answer pair that has a probability between $l$ and $h$.

For example, if we need $n = 20$ samples for each question and answer pair and set $l = 0.1$ and $h = 0.95$, then the expected number of ER samples to create is $\frac{20}{\min\{0.1,1-0.95\}} = 400$. The smaller (larger) the value of $l$ ($h$), the more samples need to be collected to guarantee enough samples per question and answer pair.

**Proposition 4.4.** Using the pre-computation technique does not alter the approximation quality of the Monte-Carlo method in Section IV-B2.

**Proof:** It is sufficient to show that for each question and answer pair $Q(a,b)$ and ans, the pre-computed ER results are based on possible worlds from the probability graph $P[a,b \rightarrow \text{ans}]$ where $P[a,b \rightarrow \text{ans}]$ is identical to $P$ except that $P[a,b \rightarrow \text{ans}](a,b)$ is set to ans. Given a question $Q$ and answer ans, say we choose an $n$-size subset of the the ER results where $Q$ was chosen (if ans = YES) or not (if ans = NO) in the possible world. Since we select edges from the probability graph $P$ independently according to their probabilities, the distribution of questions selected is identical to choosing questions from $P$ given that $Q$ is chosen (if ans = YES) or not (if ans = NO) by the principle of deferred decisions [8].

By sharing ER results, we can significantly reduce the constant factor of the complexity $O(|R|^2 \times (C(E_R) + |R|^2))$. We first compute a constant number $\frac{n}{\min\{l,1-h\}}$ of ER samples, which requires a runtime complexity of $O(C(E_R))$. Next, we iterate through each of the $D$ edges in $P$ with probabilities between $l$ and $h$, and compute ER once (Step 1, Algorithm 2) and then compute $F_1$ against the $n$ ER samples. This operation has a complexity of $O(|R|^2 \times (C(E_R) + |R|^2))$. Hence, the
total complexity is \( O(|R|^2 \times (C(E_R) + |R|^2)) \). While this complexity is the same as when we do not share ER samples, the number of ER samples we have to create reduces from \( 2 \times D \times n \) to \( \frac{n}{\min\{l, 1-h\}} \).

4) **Incremental ER:** There is still a great deal of redundant computation when running ER multiple times in Step 1 of Algorithm 2 because the similarity functions used by different ER runs are not that different. We can remove some redundant ER computation by only resolving records that may be “influenced” by the current question. For example, suppose the ER algorithm performs a transitive closure on matching records in \( R = \{a, b, c, d, e, f\} \). If \( E_R(R, F) = \{a - b, c - d, e - f\} \) and the current question is \( Q(a, c) \), then we only need to re-run ER on the set \( \{a, b, c, d\} \) instead of the entire \( R \). Only resolving a small subset of \( R \) can significantly improve the subsequent runtimes of ER. We note that not all ER algorithms can be computed incrementally. In this case, we must run ER from scratch on the entire \( R \). While the incremental ER optimization does not reduce the complexity of Algorithm 1, it significantly reduces the actual runtime of ER.

**C. Half Algorithm**

We propose a simple heuristic (called the Half algorithm) that chooses the best question by picking the edge in \( P \) with a probability closest to 0.5. Intuitively, we are choosing the most “uncertain” edge for the next question, which provides the most information when answered. However, the Half method may not return the best question that results in the highest ER accuracy. For example, suppose there is an edge \( a - b \) where \( P(a, b) = 0.5 \) and an edge \( c - d \) where \( P(c, d) = 0.1 \). Say that \( a \) and \( b \) do not match with any other records while \( c \) and \( d \) match with 10 records each. Then even if \( P(c, d) \) is further away from 0.5 than \( P(a, b) \), there is a 10% chance that the human will say YES to \( Q(c, d) \) and cluster the 20 records together, which may result in a higher expected ER accuracy (assuming the 20 records do refer to the same entity). Nevertheless, in Section VI we show that the Half algorithm can perform quite well in many scenarios.

**V. Asking Multiple Questions**

Until now, we have been asking a human one question at a time. However, we may want to ask multiple questions at a time in order to reduce latency. A straightforward extension to the GCER algorithm for asking \( k \) questions is to consider all possible \( k \) combinations of questions and compare their expected accuracies. However, this process may become very expensive for large \( k \) values. A faster heuristic is to choosing the top-k questions that have the highest expected accuracies according to the GCER algorithm. Although we are no longer choosing the best \( k \) combination of questions, we are still choosing questions that can increase the accuracy the most when asked individually. We study in Section VI-B6 how increasing \( k \) influences the ER accuracy.
VI. EXPERIMENTAL RESULTS

We compare the Half and GCER algorithms with a baseline method called Rand where questions are chosen randomly from all the record pairs. We first evaluate the three algorithms on two real datasets (Cora and Hotel) and then on synthetic datasets. Our algorithms were implemented in Java, and our experiments were run in memory on a 2.4GHz Intel(R) Core 2 processor with 4 GB of RAM.

A. Real Data Experiments

We first evaluate our algorithms on a subset of the Cora dataset, which is a publicly available list of publications. Each record contains attributes including the title and authors of a paper. The gold standard for the Cora dataset contains clusters with varying sizes. Next, we experiment on a hotel dataset provided by Yahoo! Travel where tens of thousands of records arrive from different travel sources (e.g., Orbitz.com), and must be resolved before they are shown to the users. Each record contains attributes including the name and street address of a hotel. We experimented on a subset of hotel records located in the United States. In comparison to the Cora dataset, the Hotel dataset mostly comes from two data sources that do not have duplicates within themselves. As a result, there are rarely more than two records that refer to the same hotel.

Human Gold Standard: For the Hotel dataset, we have a $C^*$ gold standard where a human expert (at Yahoo!) provided YES/NO answers to each possible hotel record pair. For the Cora dataset, however, we only have the $S^*$ gold standard, which after a global analysis tells us which publication records refer to the same entity. We cannot directly use this global information as our $C^*$ gold standard. For example, suppose that the records $r_1$, $r_2$, and $r_3$ all refer to the same entity according to $S^*$ and the pairwise similarities are $F(r_1, r_2) = 0.5$, $F(r_1, r_3) = 0.7$, and $F(r_2, r_3) = 0.9$. Although all three records should be eventually resolved to the same entity, it could be the case that a human says NO to the question $Q(r_1, r_2)$ and YES to $Q(r_2, r_3)$ and $Q(r_1, r_3)$. Notice that $t$ cannot be any smaller than 0.7 because otherwise the human will say NO to $Q(r_1, r_3)$, and only $r_2$ and $r_3$ match with each other. The threshold $t$ may differ for other clusters in the gold standard. We believe this technique models humans who do not have full knowledge of the $S^*$ gold standard.

Similarity Functions: Our similarity functions use the Jaro measure [9], which computes a string similarity ranging from 0 to 1. The Cora dataset similarity function measures the title and author similarities between two records and returns the
average similarity. The Hotel dataset similarity function measures the name and street address similarities between two records and returns the average similarity.

**ER Algorithm:** We use the Sorted Neighborhood (SN) algorithm [10] for resolving records. The SN algorithm initially sorts the records in $R$ using a certain key assuming that closer records in the sorted list are more likely to match. For example, suppose that we have the input set of records $R = \{r_1, r_2, r_3\}$ and sort the records by their names (which are not visible in this example) in alphabetical order to obtain the list $[r_1, r_2, r_3]$. The SN algorithm then slides a fixed-sized window on the sorted list of records and compares all the pairs of records that are inside the same window at any point. For each pair of records, the SN algorithm considers them to match if their similarity is above a threshold $t$. If the window size is 2 in our example, then we compare $r_1$ with $r_2$ and then $r_2$ with $r_3$, but not $r_1$ with $r_3$ because they are never in the same window. We thus produce pairs of records that match with each other. We can repeat this process using different keys (e.g., we could also sort the person records by their address values). After collecting all the pairs of records that match, we perform a transitive closure on all the matching pairs of records. For example, if $r_1$ matches with $r_2$ and $r_2$ matches with $r_3$, then we merge $r_1$, $r_2$, $r_3$ together into the ER result $\{r_1 - r_2, r_2 - r_3, r_1 - r_3\}$.

In our experiments, we sorted the Cora records according to their titles and then used a sliding window of size $W$ to compare the records using the Cora similarity function and a threshold $t$. For the Hotel records, we sorted them by their names and used the Hotel similarity function for matching records. We used a window size of $W = 20$ for both datasets. In order to optimize our code, we stored an ER result as a partition of records instead of a set of record pairs.

**Parameter Setting:** Table II shows the experimental settings used for the GCER algorithm on the Cora and Hotel datasets. The first parameter shows how many questions were asked to create the similarity-probability mappings. When choosing which questions to ask for the training, we selected record pairs that had a similarity of at least 0.8 because most pairs with similarities less than 0.8 were obviously non-matching records. For the Cora dataset, we thus extracted the pairs of records within 300 random records that had a similarity at least 0.8, resulting in 1,339 pairs used for training. Figure 6 shows the similarity to probability statistics based on simulated human answers to the selected Cora record pairs. For the Hotel dataset, most of the record pairs were non-matches, so we first sorted 5,000 U.S. Hotel records according to their names in order to group the likely-matching records together. We then selected training pairs from the first 2,000 records in the sorted list that had similarities at least 0.8. As a result, we trained on 2,141 record pairs. Figure 3 in Section III-A shows the similarity to probability mappings for the selected Hotel record pairs based on real human answers.

The next parameters are used for running GCER. We resolved 500 Cora records and 1,000 Hotel records. For both datasets, we used $n = 20$ samples for computing the expected accuracy of an ER result using Monte-Carlo methods. When pruning
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cora</td>
</tr>
<tr>
<td>Number of pairs trained</td>
<td>1,339</td>
</tr>
<tr>
<td>Number of records resolved</td>
<td>500</td>
</tr>
<tr>
<td>Number of samples</td>
<td>20</td>
</tr>
<tr>
<td>Thresholds (Low/High)</td>
<td>0.1/0.9</td>
</tr>
<tr>
<td>Accuracy threshold</td>
<td>0.9</td>
</tr>
<tr>
<td>Similarity threshold</td>
<td>0.95</td>
</tr>
</tbody>
</table>

**TABLE II**

GCER SETTING FOR REAL DATASETS

![Image of bar chart](image)

Fig. 6. Cora Training Statistics

questions, we set \( l = 0.1 \) and \( h = 0.9 \). Finally, when running GCER, we stopped asking questions once the ER accuracy exceeded a threshold of 0.9 for the Cora dataset and 0.85 for the Hotel dataset. Since we used low/high thresholds on probabilities for restricting the questions to consider, we sometimes ran out of questions to ask. The accuracy thresholds were set in order to prevent our experiments from running out of questions.

Finally when running \( SN \), we used a default similarity threshold of 0.95 for comparing the Cora records while using a threshold of 0.88 for comparing the Hotel records.

1) Cora Results: We compare the ER accuracy results for the GCER, Half, and Rand algorithms on 500 random publication records in the Cora dataset. For each question asked, we reflect the answer into the probability graph \( P \) and similarity function \( F \) and then re-compute the expected accuracies of all questions to choose the next best question. Figure 7 shows how the ER accuracy increases as more questions are asked. For example, after asking the 10\(^{th} \) question using GCER, we obtain an ER accuracy of 0.66. When no questions are asked yet, the ER result has a precision of 0.83, a recall of 0.2, and an \( F_1 \) accuracy of 0.33 for all three plots. Hence, questions that are answered are more likely to cluster records together than splitting them from each other. In order to arrive at an ER accuracy of \( c = 0.9 \), GCER asks 1,127 questions while Half asks 3,686 questions to arrive at the same accuracy. In comparison, Rand asks 50,457 questions. The results shows that both GCER and Half significantly outperform Rand asking 14–45x fewer questions. In addition, GCER outperforms Half using 3.3x fewer
questions by carefully choosing the right questions that can end up merging many records together.

We now change the $SN$ comparison threshold from its default value ($t = 0.95$) to $t = 0.86$. Compared to the previous experiment, $SN$ incorrectly clusters more Cora records. Hence, not only does GCER have to find questions to cluster records, but it also needs to find questions that split the incorrect clusters. Figure 8 compares the three algorithms in this scenario. When no questions are asked, the ER result has a precision of 0.82, a recall of 0.78, and an $F_1$ accuracy of 0.8. To arrive at our accuracy threshold of 0.9, GCER needs to ask 4,178 questions while Half asks 4,207 questions. In comparison, Rand asks 51,271 questions. Hence, both the GCER and Half algorithms outperform the Rand algorithm asking 12x fewer questions. The improvements against the Rand algorithms is thus much less than when $t = 0.95$ because splitting clusters is an expensive operation for both GCER and Half. Unlike finding records to cluster together, splitting a cluster involves finding the right set of critical questions that will be answered NO by the humans and clearly show that records in one set will not match with any records in another set. Since both GCER and Half are local search algorithms that find one best question at a time, it is hard to find the best multiple questions, which requires a global search.

2) Hotel Results: We now evaluate the GCER, Half, and Rand algorithms on 1,000 records in the Hotel dataset. Recall that we used the first 2,000 records in the sorted list of 5,000 Hotel records for training the similarity to probability mapping. Our resolution was done on the next 1,000 records in the sorted list. Figure 9 compares the three algorithms. Initially, the ER result has a precision of 0.78, a recall of 0.5, and an $F_1$ accuracy of 0.61. From the starting point, the GCER and Half algorithms ask 133 and 111 questions to arrive at an ER accuracy of 0.85. In comparison, Rand needs to ask 253,642 questions to arrive at the same accuracy. Hence, GCER and Half significantly outperform Rand by asking 1,907–2,285x fewer questions.
Unlike the Cora results, however, GCER and Half have near-identical performance. The reason is that most questions that are answered by humans end up clustering only two Hotel records, so GCER cannot find questions that are better than what Half finds. While the curves look steep, notice that the scale of the x-axis is logarithmic. The plots are actually growing linearly or tapering off.

B. Synthetic Data Experiments

We now evaluate our Crowd ER techniques using synthetic data. The main advantage of synthetic data is that they are much easier to generate for different scenarios and provide more insights into the operation of the GCER algorithm.

Table III shows the parameters used for generating the synthetic dataset $\mathcal{R}$ and the default values for the parameters. There are $s$ entities in the dataset that are distributed on a $d$-dimensional Euclidean space. The entities form the gold standard we would like to produce by running ER on $\mathcal{R}$. For each dimension, we randomly assign to $s$ one of the values in the list $[0, i, 2i, \ldots, (s-1)i]$ without replacement. As a result, any two entities have a distance of at least $i$ from each other for any dimension. For each entity $e$, the data set contains an average of $u$ records that represent that entity, where this number of duplicates forms a Zipfian distribution with an exponent of $f$. Each record $r$ generated for the entity $e$ contains $d$ attributes. For each attribute $a$, $r$ contains a value selected from a Zipfian distribution with an exponent of $g$ within the range of $[x-v, x+v]$ where $x$ is the $a$ value of $e$ and $v$ is the maximum deviation of a duplicate record’s value from its entity value.

We use a similarity function $F(a, b)$ that returns a similarity of $\min\{1, \frac{1}{\text{dist}(a,b)}\}$ where $\text{dist}(a,b)$ is the Euclidean distance between $a$ and $b$. For example, if $a$ contains the two values $[1, 2]$ and $b$ contains the values $[2, 3]$, then the Euclidean distance is $\sqrt{(2-1)^2 + (3-1)^2} = \sqrt{5}$, and the similarity is $\min\{1, \frac{1}{\sqrt{5}}\} = \frac{1}{\sqrt{5}}$. We used an ER algorithm (called $TC$) that performs a transitive closure on record pairs with similarities over the threshold $t$.

When simulating the human answers, we again assume (as in Section VI-A) that the human says YES to the minimum number of questions possible. For each cluster $c$ in the gold standard, we found the maximum similarity threshold $t$ such that performing the $TC$ algorithm on the records in $c$ using $t$ would result in all the records in $c$ clustering together.

When generating the similarity to probability mapping, we first generated a training dataset of 500 records with the default
TABLE III
PARAMETERS FOR GENERATING SYNTHETIC DATA

<table>
<thead>
<tr>
<th>Par.</th>
<th>Description</th>
<th>Val.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Data Generation</strong></td>
<td></td>
</tr>
<tr>
<td>$s$</td>
<td>Number of entities</td>
<td>200</td>
</tr>
<tr>
<td>$u$</td>
<td>Avg. number of duplicate records per entity</td>
<td>5</td>
</tr>
<tr>
<td>$f$</td>
<td>Zipfian exponent number of # duplicates</td>
<td>0.1</td>
</tr>
<tr>
<td>$d$</td>
<td>Number of attributes (dimensions) per record</td>
<td>2</td>
</tr>
<tr>
<td>$i$</td>
<td>Minimum value difference between entities</td>
<td>40</td>
</tr>
<tr>
<td>$v$</td>
<td>Maximum deviation of value per entity</td>
<td>50</td>
</tr>
<tr>
<td>$g$</td>
<td>Zipfian exponent number of deviation</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td><strong>Similarity Function</strong></td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>Record comparison threshold</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td><strong>GCER</strong></td>
<td></td>
</tr>
<tr>
<td>$r$</td>
<td>Number of record pairs for training</td>
<td>1,080</td>
</tr>
<tr>
<td>$b$</td>
<td>Number of buckets</td>
<td>30</td>
</tr>
<tr>
<td>$l$</td>
<td>Low threshold</td>
<td>0.1</td>
</tr>
<tr>
<td>$h$</td>
<td>High threshold</td>
<td>0.9</td>
</tr>
<tr>
<td>$n$</td>
<td>Number of samples</td>
<td>20</td>
</tr>
<tr>
<td>$c$</td>
<td>Accuracy threshold</td>
<td>0.9</td>
</tr>
<tr>
<td>$q$</td>
<td>Number of questions at a time</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 10. Similarity Mapping for Default Setting

parameters in Table III (except for $s$, which was set to 100). We then selected the $r = 1,080$ pairs that had similarities at least 0.016 and divided them into $b = 30$ buckets. Figure 10 shows the similarity to probability mapping for the synthetic records.

The training dataset was generated separately from the datasets that were resolved to evaluate our algorithms.

1) Number of Questions versus Accuracy: As before, we compare the GCER, Half, and Rand algorithms on how they improve the ER accuracy for each question asked. We resolve 1,000 synthetic records generated with the default setting in Table III. GCER, Half, and Rand ask 280, 771, and 257,312 questions, respectively, to generate an ER result with an accuracy
of $c = 0.9$. Hence, both GCER and Half outperform Rand by 333–919x, and GCER outperforms Half by 2.8x. Initially, the ER result had a precision of 0.99 and a recall of 0.48, so GCER was good at choosing the right questions that help cluster many records together.

2) Number of Samples Impact: We show how the number of Monte Carlo samples $n$ influences the ER accuracy of GCER. We again resolve 1,000 synthetic records generated with the default setting in Table III. Figure 12 shows the numbers of GCER questions to arrive at an ER accuracy of $c = 0.9$ when $n$ is set to 5, 20, and 100. Intuitively, a larger $n$ size enables GCER to correctly predict the expected accuracies of questions using Monte-Carlo methods. Consequently, GCER successfully finds better questions and asks fewer questions to arrive at a high ER accuracy. However, the number of questions asked when $n = 20$ (280) is not that different from when $n = 100$ (270). This result suggests that using a relatively small number of samples is sufficient for GCER to perform well.

While we are mainly interested in finding the next best question, it is also interesting to see how well GCER predicts the absolute value of the ER accuracy. That is, we would like to see how close the estimate $E[Acc(O,W)]$ is to the actual $F_1$ accuracy of $O$ against the $S^*$ gold standard. The predictions can potentially be used to measure the ER progress of GCER. For example, when $n = 20$ and we ask the 50th question, GCER predicts an ER accuracy of 0.729 while the actual ER accuracy is 0.703. In this case, GCER has over-estimated the ER accuracy by 3.7%. Throughout all the experiments above, GCER over-estimates the ER accuracy within the range of 1.4–4.2%.

3) Quality of Mapping Impact: The similarity to probability statistics is an important piece of information that guides GCER to choose the best question. We study how the number of questions asked to construct the similarity-probability
mapping influences the accuracy of GCER. We first generate three random training sets $R_1$, $R_2$, and $R_3$ with 100, 500, and 2,500 records, respectively, using the default parameters in Table III. We then choose the record pairs from $R_1$, $R_2$, and $R_3$ with similarities at least 0.016, 0.016, and 0.008, respectively. As a result, we train on 234, 1,080, and 5,743 pairs of records, respectively. Finally, we resolve 1,000 synthetic records generated with the default setting in Table III. Figure 13 shows the number of questions asked by GCER for each scenario. As the mapping becomes more precise, GCER does a better job in choosing the best question and thus asks fewer questions to arrive at a high ER accuracy.

4) Pruning Impact: Recall that pruning questions with the $l$ and $h$ thresholds reduces the number of questions to compare. In addition, setting the thresholds also reduces the expected number of ER samples to generate (see Section IV-B1). While pruning questions may improve the runtime of GCER (see Section VI-B7), we may also miss the opportunity to choose the question with the highest expected accuracy. For our experiments, we resolve 1,000 synthetic records generated with the default setting in Table III. Figure 14 evaluates GCER with the following $l$ and $h$ threshold pairs: [0.1, 0.9], [0.2, 0.8], [0.3, 0.7], and [0.4, 0.6]. Pruning is most effective for questions with probabilities that are close to 1 (0) where we can safely assume that the human will say YES (NO). As we prune more probabilities that are closer to 0.5, then we have a performance similar to the Half algorithm. If we prune too many questions, then we may run out of questions to consider. For example, unlike the first three scenarios, using the thresholds [0.4, 0.6] only results in an ER accuracy of 0.8 because we run out of questions with probabilities between 0.4 and 0.6.

5) Entity Distance Impact: Our synthetic datasets let us study scenarios where entities are closer to or further away from each other. We resolve 1,000 synthetic records generated with the default setting in Table III except for the $i$ parameter.
Figure 15 shows the GCER results when the $i$ parameter is set to the values 20, 30, 40, and 50. Recall that each entity is surrounded by a cluster of records with a diameter of $2 \times v = 100$. In addition, most of the records are concentrated at the entity in the middle because of the Zipfian exponent $g = 1$ used for distributing records within the cluster. When $i = 50$, only a few clusters that have close entities overlap with each other. As $i$ decreases, then the clusters start moving closer to each other and overlap more. As a result, there is a higher chance that records incorrectly cluster together by $TC$, and GCER spends more time splitting clusters.

6) Multiple Questions: Until now, we have only considered the problem of asking one question at a time. In practice, however, we may want to ask multiple questions to humans at a time. Instead of finding the best $k$ questions, we simply ask the top-$k$ questions that have the highest expected accuracy values according to GCER (see Section V). We study how the number of questions asked influences the ER accuracy increase per question. Figure 16 shows the GCER performance when we ask 1, 10, and 100 questions at a time. The more questions we ask at a time, the less accuracy we obtain because we lose the opportunity to reflect the human answers when finding the next set of best questions. Nevertheless, we can significantly reduce the number of invocations of finding the next best question with only a relatively small increase in the total number of questions to ask. For example, by asking 100 questions at a time instead of one, we only increase the total number of questions from 280 to 400, so we perform $\frac{100}{400/280} = 70x$ fewer invocations to arrive at the same ER accuracy.

7) Scalability: The GCER algorithm is inherently expensive. First, we need to consider potentially all possible pairs of questions, which is a quadratic operation. Next, we need to estimate the accuracy benefits of all the questions, which is also expensive. With this knowledge, we now show the scalability of GCER by increasing the number of entities $e$ from 200 to
800 for generating the synthetic data. The other parameters are set to their default values in Table III. Since we generate on average $d = 5$ duplicates per entity, we are resolving 1,000 to 4,000 records. For each $e$, we test on four different $l$ and $h$ threshold pairs: $[0.1, 0.9]$, $[0.2, 0.8]$, $[0.3, 0.7]$, and $[0.4, 0.6]$. As we can see in the Figure 17, the average time for finding the one best question increases quadratically to the number of records resolved. The increased runtime is due to the increased number of records to resolve and the quadratic complexity of $TC$. At the same time, the runtime decreases as the thresholds become stricter.

While GCER is indeed expensive, taking a few minutes to select the best question may be acceptable since the process of asking humans will often take even longer. In the case where GCER is still taking too long, we can reduce the execution time by using the following techniques (in addition to pruning more questions). First, by selecting multiple questions at once, we can amortize the cost of generating the best question. We have already seen in Section VI-B6 that asking more questions at a time significantly reduces the number of invocations of finding the next best question. Next, we can use blocking techniques [10] where $R$ is split into (possibly overlapping) blocks $R_1, \ldots, R_k$. Each $R_i$ should be small enough to search for the best questions. Only the records with the same block are compared assuming that records in different blocks are unlikely to match. For example, we might partition a set of people records according to the zip codes in address fields and only compare the records with the same zip code. The blocking techniques allow us to scale GCER on large datasets where the total number of questions increases in a linear fashion.

VII. RELATED WORK

Entity Resolution has been studied under various names including record linkage, merge/purge, deduplication, reference reconciliation, object identification, and others (see [11], [9] for recent surveys). Many ER algorithms fit into our two-phase framework of first identifying candidate matching pairs of records, and then returning the final pairs of matching records [12].

Recently, many platforms have been developed for human computation [13], [1], and several systems have been proposed for incorporating human work into a database system. CrowdDB [14] is a relational query processing system that uses microtask-based crowdsourcing to answer queries that cannot otherwise be answered. CrowdDB assumes an open-world model for
collecting data. In comparison, Deco [15] is a declarative crowdsourcing framework that uses a closed-world assumption for collecting data from humans. Qurk [16] provides an interactive environment for humans to build queries and monitor their progress. Human computation has also been used in other operations such as joins [17] as well. In comparison, our work focuses on using human computation for the specific problem of ER.

Human learning techniques have recently been proposed for ER. Reference [18] performs unsupervised learning based on humans clustering a subset of records and then applies the trained clustering algorithm to the entire dataset. Reference [19] has recently used active learning techniques for ER where the idea is to only learn the necessary information for training the ER algorithm. In comparison, our work focuses on using humans for the resolution stage of ER.

Recently, human resolution techniques for ER have been proposed as well. Reference [20] proposes a human resolution system where authors can claim their own publications. ZenCrowd [21] uses the crowd to figure out which entities in web pages refer to the same URI. A hybrid human-machine workflow [22] has been proposed to combine algorithms and human operations for ER. After automatically matching records, the likely-matching pairs of records are verified by humans using either record pairs or clusters of records as the interface. In comparison, our techniques are not constrained by a specific domain and incrementally ask the best question(s) until we arrive at a high ER accuracy instead of asking a fixed set of candidates pairs.

Active learning can be used to train similarity functions [19], [23] by selecting the best set of examples to be labeled. While selecting examples is in the same spirit as selecting questions, our work complements active learning by utilizing the crowd on top of the trained similarity functions.

VIII. Conclusion

We have proposed an ER framework that incorporates the wisdom of the crowd for resolving records. By asking humans, we are able to identify matching record pairs and thus perform better ER. We have used a probabilistic framework for predicting the human answers and estimating the expected accuracy we obtain by asking each question. Our framework is general and can compute the best question for a wide range of ER algorithms. We proved that computing the expected accuracy of a question is \#P-hard and proposed an approximation algorithm that runs in polynomial time. We have evaluated our best question algorithms on real and synthetic datasets and showed how we can obtain an ER result with high accuracy while significantly reducing the number of questions asked to humans.

References


