Question Selection for Crowd Entity Resolution

Steven Euijong Whang[*]
Stanford University
swhang@cs.stanford.edu

Peter Lofgren
Stanford University
plofgren@stanford.edu

Hector Garcia-Molina
Stanford University
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.

1. 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, 5], 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.

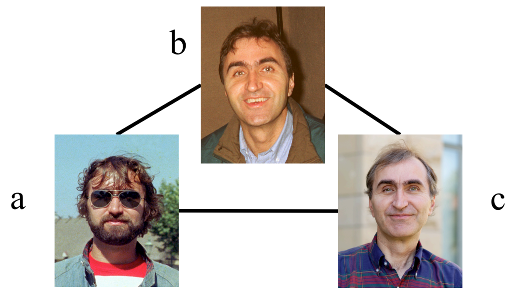

Figure 1: Resolving Photos

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 ques-

[*]current affiliation: Google Inc.

tions 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 2).
- 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 a #P-hard problem (Section 3).
- We propose three algorithms for finding the question that maximizes the expected accuracy (Section 4):
  - An exhaustive algorithm that returns the best question in exponential time.
  - The GCER (Generic Crowd Entity Resolution) algorithm that approximates the exhaustive algorithm and runs in polynomial time.
  - A simple heuristic (Half) that still gives significant improvements over a baseline method.
- We discuss how to ask multiple best questions at a time (Section 5).
- We evaluate our best question algorithms using real and synthetic datasets (Section 6).

2. PRELIMINARIES

2.1 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.

Definition 2.1. *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.

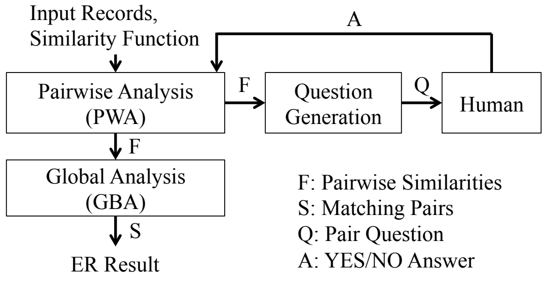

Figure 2: ER Process with Human Input

Record	Name	Address	Phone	Passport	Credit Card
a	Robert	123 Main	1234		
b	Bob	123 Main	1234	abcd	777
c	Rob			abcd	777
d	Sally		5678	efgh	999

Table 1: A Set of Person Records

To illustrate the ER process, consider the four person records in Table 1. 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 1, 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:

- $\mathcal{C}(a, c)$: Record a and c are *connected* if based on purely their contents it can be determined that they refer to the same real-world entity. In the example of Table 1, $\mathcal{C}(a, b)$ and $\mathcal{C}(b, c)$ hold, but $\mathcal{C}(a, c)$ and $\mathcal{C}(a, d)$ do not. In other words, we say that $\mathcal{C} = \{a - b, b - c\}$.
- $\mathcal{S}(a, c)$: Records a and c refer to the *same* real-world entity. In our example, $\mathcal{S}(a, b)$, $\mathcal{S}(b, c)$, $\mathcal{S}(a, c)$ hold, but $\mathcal{S}(a, d)$, $\mathcal{S}(b, d)$, $\mathcal{S}(c, d)$ do not. Hence, $\mathcal{S} = \{a - b, b - c, a - c\}$.

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

2.2 Human Interaction

We now discuss how to interact with humans in the PWA phase, as shown in Figure 2. (In Section 3, we explain how the Question Generation component chooses which questions to ask.) In particular, we may ask humans certain *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 $\mathcal{C}(a, c)$ holds, not if $\mathcal{S}(a, c)$ holds. For instance, if we ask a human to compare records a and c of Table 1, 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 $\mathcal{C}(a, c)$ 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., [12, 17]), and hence we do not believe it is necessary to discuss those issues again here.

2.3 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 $\mathcal{S}()$ set. We refer to this set as $\mathcal{S}^*()$. That is, the *final* ER output, after global analysis, should include all the record pairs in $\mathcal{S}^*()$.
- The correct answer we are given is the correct $\mathcal{C}()$ set, referred to as $\mathcal{C}^*()$. 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 $\mathcal{C}^*()$.

Note that if we are given $\mathcal{C}^*()$, we can easily compute $\mathcal{S}^*()$: That is, to generate $\mathcal{S}^*()$, we input to GBA all pairs where $\mathcal{C}^*()$ holds. (We give these pairs a similarity of 1.) The result is $\mathcal{S}^*()$. If we define $\mathcal{S}^*()$ independently without deriving it from $\mathcal{C}^*()$, we may end up defining an $\mathcal{S}^*()$ that is not achievable with our given GBA (see our technical report [19] for details).

In practice, one can have either type of gold standard. As we will see in our experiments, for one gold standard data set (Hotels), we have the correct $\mathcal{C}^*()$. (An expert compared pairs of records at a time.) For the second data set (Publications), we have a $\mathcal{S}^*()$ 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 $\mathcal{S}^*()$, one can compute a variety of metrics. For instance, precision P is the fraction of the pairs in O for which $\mathcal{S}()$ holds. Recall R is the fraction of all pairs for which $\mathcal{S}()$ holds and also appear in O. The F_1 metric is the harmonic mean of these two values, i.e., $\frac{2 \times P \times R}{P+R}$. 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.

3. 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 a #P-hard problem.

3.1 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.

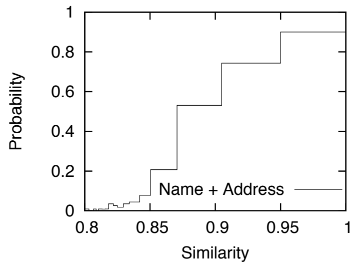

Figure 3: Hotel Training Statistics

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 $\mathcal{C}(a, c)$ 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 6 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 $\mathcal{C}^*()$ 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 $\mathcal{C}^*$ 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 $a - c \in E$ has a weight of $P(a, c)$. And since humans make no mistakes, $P(a, c)$ also gives us the probability that $\mathcal{C}^*(a, c)$ holds.

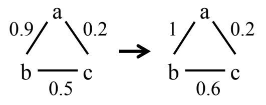

Figure 4: Similarities to Probabilities

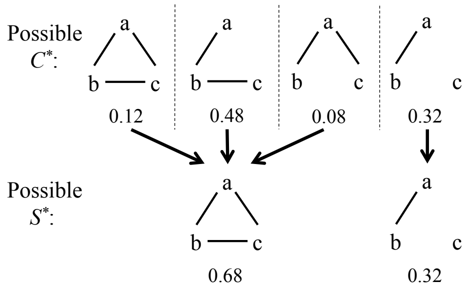

Figure 5: Possible Worlds of Figure 4

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

$$p(P, W) = (\prod_{a-b \in W} P(a, b))(\prod_{a-b \notin W} 1 - P(a, b))$$

In a similar fashion we can compute the probability of the remaining $\mathcal{C}^*$ outputs, as shown in the top half of Figure 5. For the given scenario, there are three more outputs with non-zero probability: $W_2 = \{a - b, b - c\}$ with probability 0.48, $W_3 = \{a - b, a - c\}$ 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 $\mathcal{C}^*$ *possible worlds*.

Using each $\mathcal{C}^*$ possible world W, we can compute a $\mathcal{S}^*$ 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 $\mathcal{S}^*$ possible world W' is the probability sum of each $\mathcal{C}^*$ possible world W where $W' = E_R(R, F_W)$. Continuing our example above, the bottom half of Figure 5 shows the $\mathcal{S}^*$ 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 $\mathcal{C}^*$ worlds that lead to W'_1 when transitive closure is applied. The probability of the second $\mathcal{S}^*$ world, $W'_2 = \{a - b\}$ is the probability of the fourth $\mathcal{C}^*$ world, which leads to W'_2. In a sense, the $\mathcal{S}^*$ 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 $\mathcal{C}^*$ 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 follows:

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

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 $W_1, \ldots, W_4$ in Figure 5, we can compute $E[Acc(O,W)]$ as $0.12 \times Acc(O,W_1) + 0.48 \times Acc(O,W_2) + 0.08 \times Acc(O,W_3)$ $+ 0.32 \times Acc(O,W_4) = 0.68 \times F_1(O,W_1') + 0.32 \times F_1(O,W_2')$ $= 0.68 \times \frac{2\times1\times1/3}{1+1/3} + 0.32 \times \frac{2\times1\times1}{1+1} = 0.66$.

3.3 Choosing the Best Question

Using the information in the $\mathcal{S}^*$ 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 5, 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 2.2, 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

$$\begin{aligned} E[Acc(O,W)|A(a,b)] = &P(a,b) \times E[Acc(O_1,W_1)] + \\ &(1-P(a,b)) \times E[Acc(O_2,W_2)] \end{aligned}$$

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 $\mathcal{S}^*$ 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 $\mathcal{S}^*$ possible worlds: $\{a-b, b-c, a-c\}$ with probability 0.2 and $\{a-b\}$ with probability $(1-0.2) = 0.8$. Hence, $E[Acc(O_2,W_2)] = 0.2 \times \frac{2\times1\times1/3}{1+1/3} + 0.8 \times 1 = 0.9$. 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.

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  $Z \leftarrow$ null;
2  $M \leftarrow 0$;
3  for $a-b \in \{r-s | r-s \in R \times R \wedge r \neq s\}$ do
4      $p \leftarrow P(a,b)$;
5      $t \leftarrow F(a,b)$;
6      $F(a,b) \leftarrow P(a,b) \leftarrow 1$;
7      $Y \leftarrow$ ComputeAccuracy$(R, E_R, F, P)$;
8      $F(a,b) \leftarrow P(a,b) \leftarrow 0$;
9      $N \leftarrow$ 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 $Z$;
```

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 $E[Acc(O,W)]$ is #P-hard (and thus NP-hard).*

The proof is through a reduction from the #P-hard problem of computing the probability that a random graph is connected (see our technical report [19] for details). Consequently, computing $E[Acc(O,W)]$ is #P-hard in general. The proposition also suggests that it is NP-hard to compute the optimal next question, since even evaluating how useful it is to choose a particular question is #P-hard.

4. 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 a polynomial-time approximation algorithm called GCER. Finally, we describe a simple heuristic for choosing questions without any information about the ER algorithm.

4.1 Exhaustive Algorithm

The exhaustive algorithm (Algorithm 1) iterates all the edges in P and computes the expected F_1 for asking that question. The example in Section 3.3 roughly demonstrates how Algorithm 1 would run on the similarity to probability mapping in Figure 4. In our technical report [19], we provide a step-by-step explanation on how Algorithm 1 runs on the mapping above.

We now show the time complexity of Algorithm 1. The proof can be found in our technical report [19].

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 $L \leftarrow \mathcal{C}^*$ 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 $A$;
```

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 the $\mathcal{C}^*$ possible worlds of P, and $C(E_R)$ the complexity of E_R.*

4.2 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.

4.2.1 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 6.2.4, we study the tradeoffs of pruning questions.

4.2.2 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 $E[Acc(P, O)]$. In our technical report [19], we discuss the right number of samples needed to estimate $E[Acc(P, O)]$ using Hoeffding's inequality.

By using the Monte-Carlo method, we can reduce the complexity of Algorithm 1 to be polynomial. The proof can be found in our technical report [19].

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

4.2.3 Sharing ER Results

Even with the Monte-Carlo methods in Section 4.2.2, 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 $\mathcal{C}^*$ 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 [$Q(b, c)$, YES] and [$Q(a, c)$, YES] while S_3 is a possible sample for the pairs [$Q(b, c)$, NO] and [$Q(a, c)$, 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 $\frac{n}{\min\{l, 1-h\}}$.*

The proofs for this and the following propositions are in our technical report [19].

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 4.2.2.*

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 $\frac{n}{\min\{l, 1-h\}}$.

4.2.4 *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 rerun 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 ER runtime.

4.3 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 6 we show that the Half algorithm can perform quite well in many scenarios.

5. 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 6.2.6 how increasing k influences the ER accuracy.

6. 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.

6.1 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 $\mathcal{C}^*$ 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 $\mathcal{S}^*$ 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 $\mathcal{C}^*$ gold standard. For example, suppose that the records r_1, r_2, and r_3 all refer to the same entity according to $\mathcal{S}^*$ 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 $Q(r_1, r_3)$.

We simulate the human answers for the Cora records to be as "conservative" as possible. For each cluster $c = \{r_1, r_2, \ldots, r_k\}$ in the gold standard, the human only says YES to the minimum record pairs such that running ER on c still results in all the records in c matching with each other. Continuing our example from above, suppose that the ER algorithm performs a transitive closure on pairs of records with similarity at least t. In this case, we set $t = 0.7$ where the 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 $\mathcal{S}^*$ gold standard.

Similarity Functions. Our similarity functions use the Jaro measure [20], 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 [11] 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 2 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 3.1 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 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.

6.1.1 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

Parameter	Value	
	Cora	Hotel
Number of pairs trained	1,339	2,141
Number of records resolved	500	1,000
Number of samples	20	20
Thresholds (Low/High)	0.1/0.9	0.1/0.9
Accuracy threshold	0.9	0.85
Similarity threshold	0.95	0.88

Table 2: GCER Setting for Real Datasets

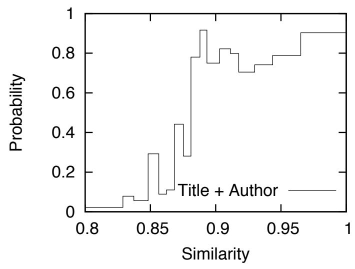

Figure 6: Cora Training Statistics

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 questions that can merge 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.

6.1.2 Hotel Results

We now evaluate the GCER, Half, and Rand algorithms

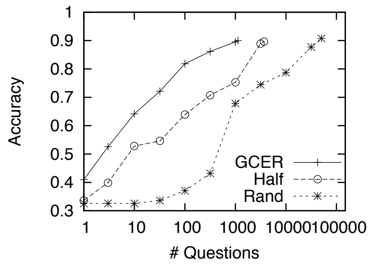

Figure 7: Cora Results using High Threshold

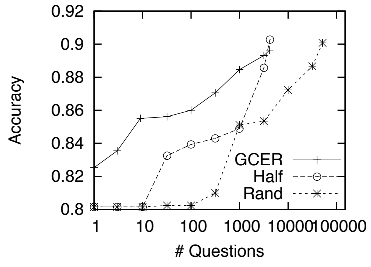

Figure 8: Cora Results using Low Threshold

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 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.

6.2 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 GCER.

Table 3 shows the parameters used for generating the synthetic dataset 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 R. For each dimension, we randomly assign to s one of the values in the list $[0, i, 2 \times i, \ldots, (s-1) \times i]$ 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

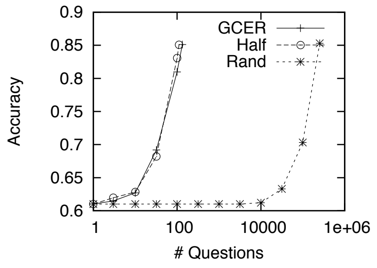

Figure 9: Hotel Results

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

Table 3: Parameters for Generating Synthetic Data

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 $\min\{1, \frac{1}{dist(a,b)}\}$ where $dist(a, b)$ 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 6.1) 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 parameters in Table 3 (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.

6.2.1 *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 3. GCER, Half, and Rand ask 280, 771, and 257,312 questions, respectively, to

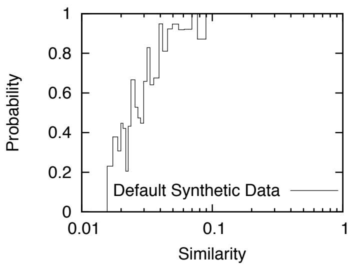

Figure 10: Similarity Mapping for Default Setting

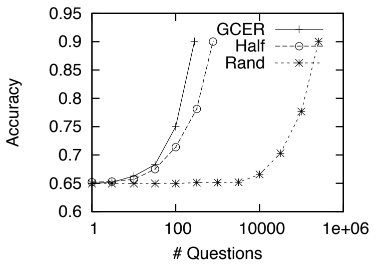

Figure 11: ER Accuracy Improvement

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.

6.2.2 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 3. 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 $\mathcal{S}^*$ gold standard. The predictions can potentially be used to measure the ER progress of GCER. For example, when $n = 20$ and we ask the 50^{th} 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%.

6.2.3 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,

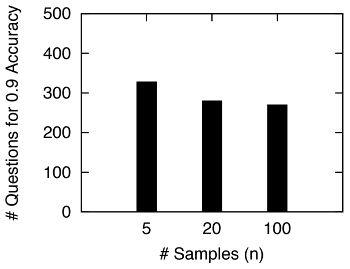

Figure 12: Sample Size Impact, GCER

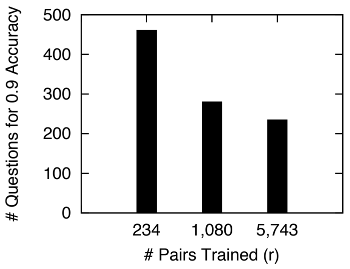

Figure 13: Training Set Size Impact, GCER

respectively, using the default parameters in Table 3. 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 3. 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.

6.2.4 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 4.2.1). While pruning questions may improve the runtime of GCER (see Section 6.2.7), 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 3. 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.

6.2.5 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 3 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

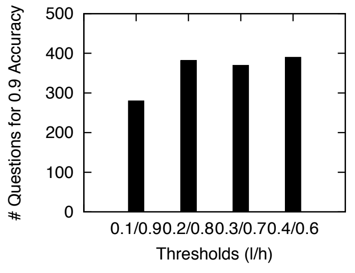

Figure 14: Pruning Impact, GCER

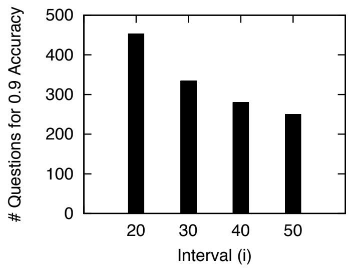

Figure 15: Interval Impact, GCER

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.

6.2.6 *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 5). 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 1, we only increase the total number of questions from 280 to 400, performing $\frac{100}{400/280} =$ 70x fewer invocations to arrive at the same ER accuracy.

6.2.7 *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 3. 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

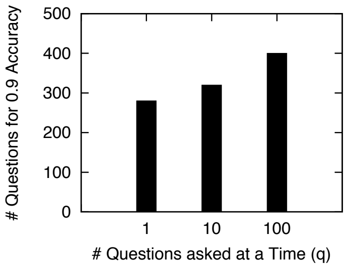

Figure 16: Asking Multiple Questions, GCER

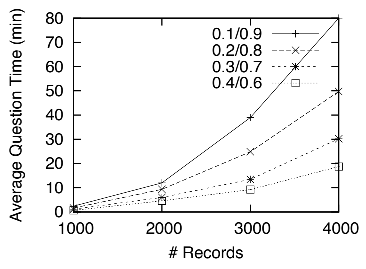

Figure 17: Scalability, GCER

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 6.2.6 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 [11] 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.

7. RELATED WORK

Entity Resolution has been studied under various names including record linkage, merge/purge, deduplication, reference reconciliation, object identification, and others (see [7, 20] 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 [10].

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 [8] 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 [16] is a declarative crowdsourcing framework that opts for more flexibility and generality. Qurk [14] 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 [15] 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 [9] performs unsupervised learning based on humans clustering a subset of records and then applies the trained clustering algorithm to the entire dataset. Reference [2] 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 [21] proposes a human resolution system where authors can claim their own publications. ZenCrowd [6] uses the crowd to figure out which entities in web pages refer to the same URI. A hybrid human-machine workflow [18] 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 [2, 4] 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.

8. 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.

9. REFERENCES

[1] Amazon mechanical turk. https://www.mturk.com.

[2] A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *SIGMOD Conference*, pages 783–794, 2010.

[3] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[4] M. Bilenko and R. J. Mooney. Employing trainable string similarity metrics for information integration. In *IIWeb*, pages 67–72, 2003.

[5] Crowdflower. http://crowdflower.com.

[6] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW*, pages 469–478, New York, NY, USA, 2012.

[7] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[8] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD Conference*, pages 61–72, 2011.

[9] R. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. In *NIPS*, 2011.

[10] O. Hassanzadeh, F. Chiang, R. J. Miller, and H. C. Lee. Framework for evaluating clustering algorithms in duplicate detection. *PVLDB*, 2(1):1282–1293, 2009.

[11] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *Proc. of ACM SIGMOD*, pages 127–138, 1995.

[12] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 64–67, New York, NY, USA, 2010.

[13] E. Law and L. von Ahn. *Human Computation.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2011.

[14] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Demonstration of qurk: a query processor for humanoperators. In *SIGMOD Conference*, pages 1315–1318, 2011.

[15] A. Marcus, E. Wu, D. RKarger, S. Madden, and R. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.

[16] H. Park, R. Pang, A. G. Parameswaran, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: A system for declarative crowdsourcing. *PVLDB*, 5(12):1990–1993, 2012.

[17] P. Venetis and H. Garcia-Molina. Quality control for comparison microtasks. In *CrowdKDD*, August 2012.

[18] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. In *PVLDB*, 2012.

[19] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. Technical report, Stanford University, available at http://ilpubs.stanford.edu:8090/1047/.

[20] W. Winkler. Overview of record linkage and current research directions. Technical report, Statistical Research Division, U.S. Bureau of the Census, Washington, DC, 2006.

[21] Y. Yang, P. Singh, J. Yao, C. man Au Yeung, A. Zareian, X. Wang, Z. Cai, M. Salvadores, N. Gibbins, W. Hall, and N. Shadbolt. Distributed human computation framework for linked data co-reference resolution. In *ESWC (1)*, pages 32–46, 2011.