Disinformation Techniques for Entity Resolution

Steven Euijong Whang*, 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 disinformation. We assume that an "agent" has some sensitive information that the "adversary" is trying to obtain. For example, a camera company (the agent) may secretly be developing its new camera model, and a user (the adversary) may want to know in advance the detailed specs of the model. The agent's goal is to disseminate false information to "dilute" what is known by the adversary. We model the adversary as an Entity Resolution (ER) process that pieces together available information. We formalize the problem of finding the disinformation with the highest benefit given a limited budget for creating the disinformation and propose efficient algorithms for solving the problem. We then evaluate our disinformation planning algorithms on real and synthetic data and compare the robustness of existing ER algorithms. In general, our disinformation techniques can be used as a framework for testing ER robustness.

Categories and Subject Descriptors

H.2.0 [**Database Management**]: General—*Security, integrity, and protection*; H.3.3 [**Information Systems**]: Information Search and Retrieval—*Clustering*

Keywords

Entity Resolution; Privacy; Disinformation

1. INTRODUCTION

Disinformation is a well-known strategy used to perturb known information by adding false information. A classic example is the Normandy landing during World War II, where the Allied forces used disinformation to make the Germans believe an attack was imminent on Pas de Calais instead of Normandy. As a result, the German forces were concentrated in Pas de Calais, which made the Normandy landing one of the turning points in the war. To present a more modern example, consider the way life insurers are predicting the life spans of their customers by piecing together health-related personal information on the Web [11]. Here the customers cannot prevent the sensitive information from leaking, as they need to give it out piecemeal to purchase items, interact with their friends, get jobs, etc. However, disinformation could be used to protect sensitive information by preventing the ER algorithm used by the insurance companies from identifying with certainty the customer's say habits or genes.

Record	Model	Pixels (M)	Price (K)
r	"C300X"	30	8
s	"C300"	20	7
t	"C200"	10	5
u	"C100"	10	1
v	"C100"	10	1

Table 1: Camera Rumor Records

We adapt disinformation to an information management setting where parts of an agent's critical information has leaked to the public. For example, suppose a camera manufacturer called Cakon is working on its latest camera product called the C300X. Although the new model is supposed to be secret, some details on the specs of the C300X may have been leaked to the public as rumors by early testers, insiders, and even competitors. Such leaks may be damaging for Cakon because customers that know about the C300X may delay their purchase of old camera models until the new camera is manufactured, potentially lowering the profits of Cakon. Or a competitor camera company may realize Cakon's strategy and develop a new camera with better specs.

It is usually very hard to "delete" public information. For example, once the information of the C300X is leaked on rumor sites, Cakon may not be able to ask the person who wrote the rumor to delete her remark. Even if the rumor was deleted, several copies of the information may remain in backup servers or other web sites.

An alternative strategy is to use disinformation techniques and add even more information to what the public (adversary) knows. Specifically, the agent generates "bogus" records such that the adversary will have more difficulty resolving the records correctly. As a result, we can effectively "dilute" the existing information.

We assume that the adversary performs an Entity Resolution (ER) operation, which is the process of identifying and merging records judged to represent the same real-world entity. In our example, the adversary can then piece together various rumors about the C300X to get a more complete picture of the model specifications, including its release date.

To illustrate how disinformation can be used, suppose there are five records r, s, t, u, and v that represent camera rumors as shown in Table 1. Suppose that the five records refer to four different camera models where the ER algorithm correctly clusters the records by producing the ER result $E_1 = \{\{r\}, \{s\}, \{t\}, \{u, v\}\}$. Here, we use curly brackets to cluster the records that refer to the same entity. The ER result says that the adversary considers u and v to

*Current Affiliation: Google Inc.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://–enter the whole DOI string from rightsreview form confirmation.

refer to the same camera model while considering the remaining three records to be different models.

Now say that the agent's sensitive information is $\{r\}$, which is the target cluster that refers to the new camera model C300X (which in reality will have 30M pixels and sell for 8K dollars). The agent can reduce what is known about the C300X by generating a record that would make the adversary confuse the clusters $\{r\}$ and $\{s\}$. Generating the disinformation record would involve creating a model number that is similar to both C300X and C300 and then taking some realistic number of pixels between 20M and 30M and a price between 7K and 8K dollars. Suppose that the agent has generated the disinformation record d_1: {⟨Model, "C300X"⟩, ⟨Pixels, 20⟩, ⟨Price, 7⟩}. As a result, d_1 may match with r because they have the same model name. The ER algorithm can conceptually merge the two records into $r + d_1$: {⟨Model, "C300X"⟩, ⟨Pixels, 20⟩, ⟨Pixels, 30⟩, ⟨Price, 7⟩, ⟨Price, 8⟩} where the '+' operation denotes the merged result of two records. Now $r + d_1$ and s are now similar and might match with each other because they have the same number of pixels and price. As a result, $r + d_1$ and s may merge to produce the new ER result $E_2 = \{\{r, s, d_1\}, \{t\}, \{u, v\}\}$. While r and s were considered different entities in E_1, they are now considered the same entity in E_2 with the disinformation record d_1.

As a result of the disinformation, the adversary is confused about the upcoming C300X: Will it have 20M or 30M pixels? Will it sell for 7K or 8K dollars? With all the uncertainty, the adversary may be even less confident that the C300X is a real upcoming product. We could further reduce the correctness of the target cluster by merging it with even more clusters, and creating disinformation now becomes an optimization problem where we would like to maximize the "confusion" around an entity as much as possible using a total cost for creating the disinformation records within a fixed budget.

A practical application of our disinformation techniques is evaluating the "robustness" of ER algorithms against disinformation. That is, some ER algorithms may be more susceptible to merging unrelated records while others may still produce similar ER results.

In summary, the main contributions of this paper are:

- We formalize the notion of disinformation in an ER setting. We also define a disinformation optimization problem that maximizes confusion for a given disinformation budget. We analyze the hardness of this problem and propose a desirable ER property that makes disinformation more effective (Section 2).
- We propose efficient exact and approximate algorithms for a restricted version of the disinformation problem. We then propose heuristics for the general disinformation problem (Section 3).
- We experiment on synthetic and real data to demonstrate the effectiveness of disinformation and compare the robustness of existing ER algorithms (Section 4).

2. FRAMEWORK

2.1 ER Model

We assume a database of records $R = \{r_1, r_2, \ldots, r_n\}$. The database could be a collection of rumors, homepages, tuples, or even tweets. Each record r is a set of attributes, and each attribute can be thought of as a label and value pair, although this view is not essential for our work. We do not assume a fixed schema because records can be from various data sources that use different attributes. As an example, the following record may refer to the camera model C300X:

$$r = \{\langle \text{N, "C300X"} \rangle, \langle \text{X}, 30 \rangle, \langle \text{P}, 8 \rangle\}$$

Each attribute $a \in r$ is surrounded by angle brackets and consists of one label $a.lab$ and one value $a.val$.

An ER algorithm E takes as input a set of records and groups together records are believed to represent the same real world entity. We represent the output of the ER process as a partition of the input.

2.2 Agent's Actions

The agent knows the current public information R and assumes the adversary is using a particular algorithm E. To decide what disinformation records to add to R, the agent proceeds in two steps: (1) The agent examines the clusters in $E(R)$ and selects two of them to merge, c_i and c_j. (2) The agent uses a function $PLAN(c_1, c_j)$ that generates the disinformation records. After the disinformation records are disseminated, R becomes $R' = R \cup PLAN(c_1, c_j)$, and in $E(R')$ old clusters c_1 and c_2 appear as a single cluster. The agent can repeat these two steps on R' if necessary. The $PLAN$ function is application specific, and may involve the creation of "fake but believable" attributes and records [12]. The creation and dissemination of the disinformation records has a cost, which we model by function $D(c_i, c_j)$. We assume that this cost function is non-negative and commutative. That is, $\forall i, j, D(c_i, c_j) \geq 0$ and $D(c_i, c_j) = D(c_j, c_i)$. We assume that the merging costs for different pairs of clusters are independent of each other.

2.3 Disinformation Problem

When the agent selects clusters to merge, its goal is to maximize the "confusion" (see below) of one particular entity e, which we call the *target entity*. We call the cluster $c_0 \in E(R)$ that represents the information of e the *target cluster*. Intuitively, by merging other clusters in $E(R)$ to the target cluster, the agent can dilute the information in the target cluster and thus increase the confusion. In our motivating example, the camera company Cakon was increasing the confusion on the target entity C300X by merging the C300 cluster $\{s\}$ to the target cluster $\{r\}$. If there are multiple clusters that represent e, we choose the cluster that "best" represents e and set it as the target cluster c_0. For example, we could define the best cluster as the one containing the largest number of records that refer to e.

The confusion of target entity e is an application-specific measure that compares the true values of e's attributes to those in c_0. For example, we can define the confusion of e as the number of incorrect attributes in c_0 minus the number of correct attributes of c_0 where we count duplicate attributes. The amount of confusion we gain whenever we merge a cluster $c_i \in E(R)$ with the target cluster c_0 can be captured as the *benefit* of c_i, which is computed as $N(c_i)$ using a benefit function N. In our example above, we can define the benefit of c_i to be the number of incorrect attributes in c_i about e. Suppose that e can be represented as the record r = {⟨Model, "C300X"⟩, ⟨Pixels, 30⟩}. Then a cluster c containing the records s = {⟨Model, "C300X"⟩, ⟨Pixels, 20⟩} and t = {⟨Model, "C200"⟩, ⟨Pixels, 20⟩} has one correct attribute (i.e., ⟨Model, "C300X"⟩) and three incorrect attributes (i.e., one ⟨Model, "C200"⟩ and two ⟨Pixels, 20⟩'s). As a result, the benefit of c is 3. As a default, we always define the benefit $N(c_0)$ to be 0 because c_0 does not need to merge with itself.

Given the agent's knowledge on the ER algorithm E, database R, cost function D, and the benefit function N, we now define an optimization problem of producing the best set of pairwise cluster merges that can maximize the total benefit while using a total cost for merging clusters within a fixed budget. We first draw an undirected cost graph G among the clusters in $E(R)$ where each edge between the clusters c_i and c_j (denoted as c_i–c_j) has a weight of

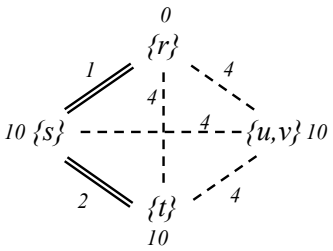

Figure 1: Cost Graph

$D(c_i, c_j)$. We denote the set of vertices in G as $G.V$ and the set of edges as $G.E$. For example, suppose that $E(R)$ = {{r}, {s}, {t}, {u, v}} and the target cluster c_0 = {r}. Also suppose that $D(\{r\}, \{s\})$ = 1, $D(\{s\}, \{t\})$ = 2, and the rest of the edges have the weight 4. In this example, we also assume that the benefits for all clusters have the value 10, except for c_0, which has a benefit of 0. The resulting cost graph G is shown in Figure 1 (ignore the double lines for now).

We view any subtree J in G that has the target cluster c_0 as its root a *disinformation plan* of the entity e that specifies which pairs of clusters should be merged together through disinformation. Just like in G, we denote the set of vertices in J as $J.V$ and the set of edges in J as $J.E$. The cost of merging the clusters connected by J is then $\sum_{(c_i,c_j)\in J.E} D(c_i, c_j)$. Continuing our example above, suppose the subtree J of G connects the three clusters {r}, {s}, and {t} with the two edges {r}–{s} and {s}–{t}. Here, the plan is to add disinformation records between {r} and {s}, and between {s} and {t} to merge the three clusters. As a result, the total merging cost of J is 1 + 2 = 3, and the total benefit obtained is 0 + 10 + 10 = 20. Figure 1 depicts the plan J by drawing double lines for the edges in J. If the subtree J instead contained the edges {r}–{s} and {r}–{t}, then the total merging cost would be 1 + 4 = 5 (but the benefit would be the same, i.e., 20).

Given a budget B that limits the total cost of generating disinformation, we define the optimal disinformation plan of e as follows.

DEFINITION 2.1. *Given a cost graph G, a target cluster c_0, a cost function D, a benefit function N, and a cost budget B, the* optimal disinformation plan *J is the subtree of G that contains c_0 and has the maximum total benefit $\sum_{c_i\in J.V} N(c_i)$ subject to $\sum_{(c_i,c_j)\in J.E} D(c_i, c_j) \leq B$.*

Using the cost graph in Figure 1, suppose that c_0 = {r} and the cost budget B = 3. As a result, the subtree J with the largest benefit connects the clusters {r}, {s}, and {t} with the edges {r}–{s} and {s}–{t} and has a total benefit of 0 + 10 + 10 = 20 and a total merging cost of $D(\{r\}, \{s\}) + D(\{s\}, \{t\})$ = 1 + 2 = 3 $\leq$ B. Merging {u, v} to c_0 will require a total merging cost of 4, which exceeds B.

A disinformation plan provides a guideline for creating disinformation. Since we assume that all the merging costs are independent of each other, a disinformation plan satisfying Definition 2.1 does not necessarily lead to an optimal disinformation in the case where the costs are not independent. However, the independence assumption allows us to efficiently find out which clusters should be merged in order to increase the confusion significantly. In Section 4 we will study the effectiveness of disinformation plans based on the independence assumption in scenarios where the merging costs are not independent.

We now show that the disinformation problem is NP-hard in the strong sense [3], which means that the problem remains NP-hard even when all of its numerical parameters are bounded by a polynomial in the length of the input. In addition, it is proven that a problem that is NP-hard in the strong sense has no fully polynomial-time approximation scheme unless P = NP. The proofs for the complexity of the disinformation problem can be found in our technical report [15].

PROPOSITION 2.2. *Finding the optimal disinformation plan (Definition 2.1) is* NP*-hard in the strong sense.*

Given that the disinformation problem is NP-hard in the strong sense, we now consider a more restricted version of the disinformation problem where we only consider disinformation plans that have heights of at most h. Here, we define the height of a tree as the length of the longest path from the root to the deepest node in the tree. For example, if a tree has a root node and two child nodes, the height is 1. We can prove that even if h = 2, the disinformation problem is still NP-hard in the strong sense. However, if h = 1 where all the clusters other than c_0 can only be directly connected to c_0 in a disinformation plan, the disinformation problem is NP-hard in the weak sense [3] where there exists a pseudo-polynomial algorithm that returns the optimal disinformation plan.

PROPOSITION 2.3. *Finding the optimal disinformation plan (Definition 2.1) with $h = 1$ is* NP*-hard in the weak sense.*

The disinformation problem with h = 1 is interesting because it captures the natural strategy of comparing the target entity e with one other entity at a time, making it a practical approach for disinformation. In Section 3, we show there are an exact pseudo-polynomial algorithm and an approximate polynomial-time algorithm for the h = 1 problem. In Section 4, we show that disinformation plans with h = 1 perform just as good as general disinformation plans in terms of maximizing the confusion of e while taking much less time to generate.

2.4 Monotonicity

We now define a property of an ER algorithm that makes our disinformation techniques more effective.

DEFINITION 2.4. *An ER algorithm E is* monotonic *if for any database R and disinformation record d, $\forall c_i \in E(R)$, $\exists c_j \in E(R \cup \{d\})$ where $c_i \subseteq c_j$.*

For example, say that the ER algorithm E is monotonic and $E(R)$ = {{r, s}, {t}}. Then if we add a disinformation record d and compute $E(R \cup \{d\})$, the records r and s can never split. Thus a possible ER result would be {{r, s, d}, {t}}, but not {{r, d}, {s}, {t}}.

The monotonicity property is helpful in the agent's point of view because we do not have to worry about the ER algorithm splitting any clusters when we are trying to merge two clusters. As a result, the analysis of the cost graph is accurate, and the agent can better predict how the ER result would change if we add disinformation records according to the optimal disinformation plan.

3. PLANNING ALGORITHMS

We start by proposing an algorithm that returns an optimal disinformation plan (Definition 2.1) where h = 1. Restricting h to 1 gives us the insight for solving the general problem later on. We propose a pseudo-polynomial algorithm that uses dynamic programming and runs in $O(|G.V| \times B)$ time, which is polynomial to the numerical value of the budget B, but still exponential to the length of the binary representation of B. We assume that B is an integer and that all the edges in the cost graph G have integer values. Next, we propose a 2-approximate greedy algorithm that runs in $O(|G.V| \times log(|G.V|))$ time. Finally, we propose two heuristics for the general disinformation problem based on the first two algorithms for the restricted problem.

3.1 Exact Algorithm for 1-Level Plans

The exact algorithm for 1-level plans uses dynamic programming to solve the disinformation problem where $h = 1$. This algorithm is similar to a dynamic algorithm used to solve the 0–1 Knapsack problem and is described in detail in our technical report [15]. Given the cost graph G, the root node $c_0 \in G.V$, the cost function D, the benefit function N, and the budget B, we first assign sequential ids starting from 1 to the vertices other than c_0 in G. Each subproblem in $\{(i, t)|\ i = 0, \ldots, |G.V| - 1$ and $t = 0, \ldots, B\}$ is defined as solving the disinformation problem for a subgraph of G that contains all the vertices up to the id i along with the edges among those vertices while using the cost budget t. We use a 2-dimensional array m where $m[i, t]$ contains the maximum benefit for each subproblem (i, t). In addition, we store the clusters in the optimal disinformation plan for each subproblem in the array s. After running the algorithm, the optimal disinformation plan J has a total benefit of $m[|G.V| - 1, B]$.

The proof of the correctness and complexities of the exact algorithm (and the following algorithms) can be found in our technical report [15].

PROPOSITION 3.1. *The exact algorithm generates the optimal disinformation plan with $h = 1$.*

PROPOSITION 3.2. *The time complexity of the exact algorithm is $O(|G.V| \times B)$, and the space complexity is $O(|G.V|^2 \times T)$.*

3.2 Approximate Algorithm for 1-Level Plans

We now propose a 2-approximate greedy algorithm that runs in polynomial time. The algorithm is similar to a 2-approximation algorithm that solves the 0–1 Knapsack problem. We first add c_0 to the disinformation plan. We then select the clusters where $D(c_0, c_i) \leq B$ and sort them by the benefit-per-cost ratio $\frac{N(c_i)}{D(c_0,c_i)}$ in decreasing order into the list $[c'_1, \ldots, c'_n]$. We then iterate through the sorted list of clusters and add each cluster to the disinformation plan J until the current total cost exceeds B. Suppose that we have added the sequence of clusters $[c'_1, \ldots, c'_k]$ where $k \leq n$. If $k = n$ or $\sum_{i=1,\ldots,k} N(c_i) > N(c_{k+1})$, we return the disinformation plan J where $J.V = \{c_0, c'_1 \ldots, c'_k\}$ and $J.E = \{c_0\text{–}c'_i | i = 1, \ldots, k\}$. Otherwise, we return the plan J where $J.V = \{c_0, c'_{k+1}\}$ and $J.E = \{c_0\text{–}c'_{k+1}\}$.

PROPOSITION 3.3. *The greedy algorithm generates a 2 approximate optimal disinformation plan with $h = 1$.*

PROPOSITION 3.4. *The time complexity of the greedy algorithm is $O(|G.V|^2 \times log(|G.V|))$, and the space complexity is $O(|G.V|)$.*

3.3 Heuristics for General Plans

Since the general disinformation problem (Definition 2.1) is NP-hard in the strong sense, there is no exact pseudo-polynomial algorithm or approximate polynomial algorithm for the problem. Instead, we propose two heuristics that extend the algorithms in Sections 3.1 and 3.2 to produce disinformation plans with no restriction in the heights. The full description of the algorithms can be found in our technical report [15].

The first heuristic (called EG) repeatedly calls the exact algorithm in Section 3.1 for constructing each level of the disinformation plan. As a result, the EG algorithm always returns a disinformation plan that is at least as good as the best 1-level plan.

PROPOSITION 3.5. *The time complexity of EG is $O(|G.V|^2 \times B + |G.V|^3)$, and the space complexity is $O(|G.V|^2 \times B)$.*

Algorithm	Description
$E2$	Exact algorithm for 1-level plans
EG	Heuristic extending $E2$ for general plans
$A2$	Greedy algorithm for 1-level plans
AG	Heuristic extending $A2$ for general plans

Table 2: Disinformation Plan Algorithms

Our second heuristic (called AG) extends the greedy algorithm in Section 3.2. Again, we first sort the clusters other than c_0 that have a cost $D(c_0, c_i) \leq B$ by their $\frac{N(c_i)}{D(c_0,c_i)}$ values in decreasing order. The algorithm then only merges the cluster with the highest benefit-per-cost ratio to the closest cluster in the current plan and updates the edges and the budget just like in the EG algorithm. We repeat the process of sorting the remaining clusters and merging the best one with the closest cluster in the plan until no cluster can be be merged without costing more than the budget.

PROPOSITION 3.6. *The time complexity of AG is $O(|G.V|^2 \times log(|G.V|))$, and the space complexity is $O(|G.V|)$.*

4. EXPERIMENTS

We evaluate the disinformation planning algorithms in Section 3 (summarized in Table 2) on synthetic data (Section 4.1) and then on real data (Section 4.2). We compare the robustness of two ER algorithms in the literature: Single-link Hierarchical Clustering [5, 7] (HC) and Sorted Neighborhood [4] (SN). Details on the ER algorithms, the confusion metric, and the benefit and cost functions can be found in our technical report [15].

4.1 Synthetic Data Experiments

We evaluate our disinformation 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 our planning algorithms. The details for generating the synthetic data, setting the target entity and cluster, and generating disinformation can be found in our technical report [15]. Although not presented here due to space restrictions, we also show in our technical report [15] how the disinformation algorithms perform with partial information, restrictions on creating values, higher-dimensional data, and larger data.

4.1.1 ER Algorithm Robustness

We compare the robustness of the HC and SN algorithms against the $E2$ planning algorithm. (Using any other planning algorithm produces similar results.) We vary the budget B from 100 to 400 records and see the increase in confusion as we generate more disinformation records. Since we choose the target entity as the one with the largest number of duplicates, it takes many disinformation records to significantly increase the confusion. For target entities with fewer duplicates, the increase of confusion is much more rapid (see Section 4.1.4).

Figure 2 shows that the overall confusion results for the SN algorithm are lower than those of the HC algorithm. Initially, the ER results without the disinformation were nearly the same where the SN algorithm produced 105 clusters with the largest cluster of size 195 while the HC algorithm produced 104 clusters with the largest cluster of size 196. However, as we add disinformation records, the SN algorithm shows a much slower increase in confusion, demonstrating that it is more robust to disinformation than the HC algorithm. The main reason is that HC satisfies monotonicity, so clusters are guaranteed to merge by adding disinformation whereas the SN algorithm may not properly merge the same clusters despite the disinformation.

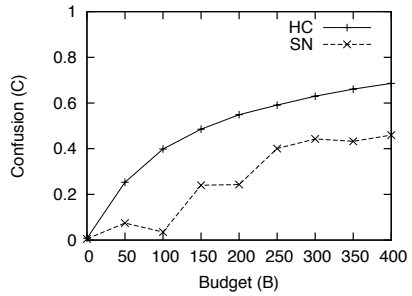

Figure 2: Robustness of the HC and SN algorithms

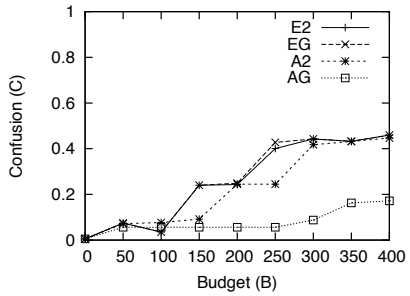

Figure 3: Comparison of disinformation algorithms

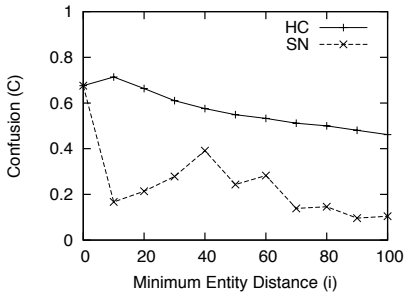

Figure 4: Entity distance impact on confusion

Figure 3 compares the four planning algorithms using the SN algorithm. We can observe in the figure that the EG, $E2$, and $A2$ algorithms have similar confusion results. Interestingly, the AG algorithm performs consistently worse than the other three algorithms when the budget exceeds 100. The reason is that the AG algorithm was generating disinformation plans with large heights (e.g., the optimal plan when B = 200 had a height of 8), but the SN algorithm was not able to merge all the clusters connected by the plan due to the limited sliding window size (used by SN to limit the number of records compared). For example, even if two clusters c_1 and c_2 were connected with a straight line of disinformation records, the records of some other cluster c_3 were preventing some of the records connecting c_1 and c_2 from being compared within the same sliding window.

4.1.2 Entity Distance Impact

We investigate how the distances among entities influence the confusion results. Figure 4 shows how the accuracies of the HC and SN algorithms change depending on the minimum distance i between entities using a budget of B = 200 records. The closer the entities are with each other (i.e., as i decreases), the more likely the ER algorithm will mistakenly merge different clusters, which leads to a higher confusion. The HC algorithm plot clearly shows this trend. The only exception is when i decreases from 10 to 0. The confusion happens to slightly decrease because some of the records that were newly merged with the target cluster were actually correct records that referred to e. The SN algorithm plot becomes increasingly unpredictable as i decreases. The reason is that when merging two clusters with disinformation, there is a higher chance for other clusters to interfere with the disinformation.

4.1.3 Universality of Disinformation

In practice, the agent may not be able to tell which ER algorithm the adversary will use on her database. Hence, it is important for our disinformation techniques to be universal in a sense that the disinformation records generated from the agent's ER algorithm should increase the confusion of the target entity even if the adversary uses any other ER algorithm. We claim that, as long as the ER algorithm used for generating the disinformation "correctly" clusters the records in the database, the optimal disinformation generated by using the agent's ER algorithm are indeed applicable when the adversary uses a different ER algorithm.

Figure 5 shows the results of using the disinformation generated when the agent assumes the SN (HC) algorithm while the adversary actually uses the HC (SN) algorithm. We observe that there is almost no change in the confusion results compared to when the agent and adversary use the same ER algorithms. The reason is that the HC and SN algorithms identified nearly the same entities when resolving R, so the generated disinformation records were nearly the same as well.

4.1.4 Target Entities with Fewer Duplicates

In this section, we consider target entities that have fewer duplicates and observe how their confusion values increase against disinformation. The fewer the duplicates, the more rapidly the confusion increases as a result of merging clusters. For example, suppose that Cakon has made an official announcement of a new camera model. With a lot of press coverage (i.e., there are many duplicate records about the model), it is hard to confuse the adversary of this information even with many false rumors. However, if Cakon has not made any announcements, and there are only speculations about the new model (i.e., there are few duplicate records), then it is much easier to confuse the adversary by adding just a few false rumors. Figure 6 shows the confusion results when we use the entities with the k-th most duplicates as the target entities where k varied from 1 to 50. As a result, the entities with fewer duplicates tend to have a more rapid increase in confusion against the same budget. For example, we only need to generate 3 disinformation records to increase the confusion of the entity with the 50-th largest number of duplicates to 0.53. Our results show that it is easier to confuse the adversary on entities with fewer duplicates.

4.2 Real Data Experiments

We now evaluate our disinformation techniques on real data to see how disinformation works in two domains where records are not necessarily in a Euclidean space. Suppose that a celebrity wants to hold an event in a secret location without letting the public know. She might want to confuse the adversary by creating false information about locations. Using this scenario, we experimented on a hotel database where the hotel records simulate possible locations for the secret event. The hotel data was 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 hotel record contains a name, street address, city, state, zip code, and latitude/longitude coordinates. We experimented on a random subset of 5,000 hotel records located in the United States. Resolving hotel records involves the comparison of multiple non-Euclidean values. Details on matching records and generating disinformation can be found in our technical report [15].

In Figure 7, we evaluate the four disinformation planning algorithms on the hotel records. Here, the target cluster only had a size of 3, so the confusion of the target entity was sensitive to even a few records merging with the target cluster. For example, using 10 disinformation records, the confusion of the target entity increased to 0.4. The four planning algorithms produce identical confusion results as the budget increases because the cluster sizes were very

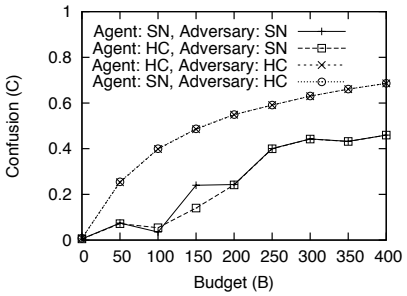

Figure 5: Universal disinformation

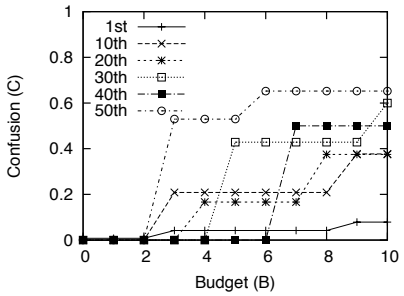

Figure 6: Entities with fewer duplicates

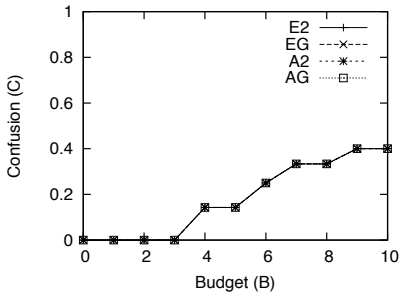

Figure 7: Hotel data confusion

uniform, so there was little incentive to use multi-level plans so that "far away" clusters would merge with the target cluster. The results show that, even if we generate disinformation on a non-Euclidean space, we were still able to significantly increase the confusion for a complex adversary ER algorithm.

5. RELATED WORK

Entity Resolution has been studied under various names including record linkage, merge/purge, deduplication, reference reconciliation, object identification, and others (see [16, 2] for recent surveys). Most work focuses on improving the ER quality or scalability. In contrast, our approach is to dilute the information of ER results by adding disinformation records. Our techniques can be useful when sensitive information has leaked to the public and cannot be deleted.

The problem of managing sensitive information in the public has been addressed in several works. The P4P framework [1] seeks to contain illegitimate use of personal information that has already been released to an adversary. For different types of information, general-purpose mechanisms are proposed to retain control of the data. Measures based on ER [13, 14] have been proposed to quantify the amount of sensitive information released to the public. Reference [6] defines the leakage of information in a general data mining context and provides detection and prevention techniques for leakage. In comparison, our work models the adversary as an ER operator and maximizes the confusion of the target entity.

A recent line of work uses disinformation for managing sensitive information in the public. Reference [8] uses disinformation while distributing data to detect if any information has leaked and to tell who was the culprit. Reputation.com [9] uses disinformation techniques for managing the reputation of individuals on the Web. For instance, Reputation.com suppresses negative information of individuals in search engine results by creating new web pages or by multiplying links to existing ones. TrackMeNot [10] is a browser extension that helps protect web searchers from surveillance and data-profiling by search engines using noise and obfuscation. In comparison, our work uses disinformation against an ER algorithm to increase the confusion of the target entity.

6. CONCLUSION

Disinformation is an effective strategy for an agent to prevent an adversary from piecing together sensitive information in the public. In addition, disinformation can be used to evaluate the robustness of ER algorithms. We have formalized the disinformation problem by modeling the adversary as an ER process and proposed efficient algorithms for generating disinformation that induces the target cluster to merge with other clusters. Our experiments on synthetic data show that the optimal disinformation can significantly increase the confusion of the target entity, especially if the ER algorithm satisfies monotonicity. We have shown that the optimal disinformation generated from correct ER results can be applied when the adversary uses a different (but correct) ER algorithm. Also, our techniques are more effective when there are fewer duplicates of the target entity. Finally, we have demonstrated with real data that our disinformation techniques are effective even when the records are not in a Euclidean space, and the match function is complex.

7. REFERENCES

[1] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, N. Mishra, R. Motwani, U. Srivastava, D. Thomas, J. Widom, and Y. Xu. Vision paper: Enabling privacy for the paranoids. In *VLDB*, pages 708–719, 2004.

[2] P. Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer, 2012.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

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

[5] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[6] S. Kaufman, S. Rosset, and C. Perlich. Leakage in data mining: formulation, detection, and avoidance. In *KDD*, pages 556–563, 2011.

[7] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[8] P. Papadimitriou and H. Garcia-Molina. Data leakage detection. *IEEE TKDE*, 23(1):51–63, 2011.

[9] Reputation.com. http://www.reputation.com.

[10] TrackMeNot. http://cs.nyu.edu/trackmenot.

[11] Wall Street Journal. Insurers test data profiles to identify risky clients, 2011.

[12] S. E. Whang. *Data Analytics: Integration and Privacy*. PhD thesis, Stanford University, 2012.

[13] S. E. Whang and H. Garcia-Molina. Managing information leakage. In *CIDR*, pages 79–84, 2011.

[14] S. E. Whang and H. Garcia-Molina. A model for quantifying information leakage. In *SDM*, pages 25–44, 2012.

[15] S. E. Whang and H. Garcia-Molina. Disinformation techniques for entity resolution. Technical report, Stanford University, available at http://ilpubs.stanford.edu:8090/1014/.

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