Selecting Data to Clean for Fact Checking: Minimizing Uncertainty vs. Maximizing Surprise*

Stavros Sintos, Pankaj K. Agarwal, and Jun Yang
Department of Computer Science, Duke University
{ssintos, pankaj, junyang}@cs.duke.edu

ABSTRACT

We study the optimization problem of selecting numerical quantities to clean in order to fact-check claims based on such data. Oftentimes, such claims are technically correct, but they can still mislead for two reasons. First, data may contain uncertainty and errors. Second, data can be "fished" to advance particular positions. In practice, fact-checkers cannot afford to clean all data and must choose to clean what "matters the most" to checking a claim. We explore alternative definitions of what "matters the most": one is to ascertain claim qualities (by minimizing uncertainty in these measures), while an alternative is just to counter the claim (by maximizing the probability of finding a counterargument). We show whether the two objectives align with each other, with important implications on when fact-checkers should exercise care in selective data cleaning, to avoid potential bias introduced by their desire to counter claims. We develop efficient algorithms for solving the various variants of the optimization problem, showing significant improvements over naive solutions. The problem is particularly challenging because the objectives in the fact-checking context are complex, non-linear functions over data. We obtain results that generalize to a large class of functions, with potential applications beyond fact-checking.

PVLDB Reference Format:
Stavros Sintos, Pankaj K. Agarwal, Jun Yang. Selecting Data to Clean for Fact Checking: Minimizing Uncertainty vs. Maximizing Surprise. *PVLDB*, 12(13): 2408 - 2421, 2019.
DOI: https://doi.org/10.14778/3358701.3358708

1. INTRODUCTION

We proud ourselves in basing our decisions on data and evidence, yet "data determinism" is not without its own issues. Two glaring issues are data quality—where inaccurate data leads to incorrect conclusions—and the practice of *data fishing*—where, even when assuming perfect data, one can cherry-pick data to make correct but still misleading claims. Journalists and fact-checkers devote an enormous amount of effort to checking claims based on data, and have to struggle constantly with both issues, often with limited time and resources. Consider the following example.

EXAMPLE 1. *Our world is never short of controversial and contradictory claims about crime statistics.* "There have been huge drops in the murder rates in cities." "Neighborhoods have become more violent under his watch." *The list goes on. While we have left out the specifics here, a few internet searches will reveal a long list of such claims, all seemingly backed by data.*

Fact-checking such claims is no easy task. First, there are numerous well-documented data quality issues with crime data [3, 32]. In the U.S., a primary source for such data is the Uniform Crime Reports*, which relies on voluntary reporting from law enforcement agencies at the different levels of jurisdiction. Coding errors and inconsistencies, changes to reporting guidelines, or even reporting delays and personnel changes can lead to under- or overstatement of crimes for particular jurisdictions in particular time periods. To "clean" a potentially erroneous value, a fact-checker may need to go through local agencies and/or consult with experts who have previously worked with the data. With limited time and resources, it is often impractical to clean all relevant data items.*

Second, many claims are correct but misleading. For example, a claim about rising or falling crime rate may be made over a particular time period. However, if we simply change the period a bit, the trend may become less pronounced or even reversed. In that case, the claim is not fair or robust, and we can show a "counter" claim, similar in form to the original but with a different conclusion, to help refute the original. As another example, if the claim is intended as an attack on the leadership for a particular jurisdiction, we can check whether similar claims can be made for other jurisdictions or for other previous leaderships. If yes, then the original claim is not unique. Fact-checkers frequently employ such analyses in rebutting correct but misleading claims, e.g. [3, 17, 37].

To combat the problems of data quality and data fishing, we can draw methods from data cleaning [16, 21] and recent work on fact-checking correct but misleading claims [43, 44]. This paper explores the following specific question: given a limited budget to fact-check a claim, which data items should we choose to clean? Intuitively, we would like to prioritize efforts towards those parts of data that "matter the most" to fact-checking the given claim.

Consider claims that can be modeled as queries over a database. When the values in the database are uncertain, the correctness of the claim (query result) is uncertain too. We can spend some budget to remove uncertainty in some data values, which can help reduce uncertainty in claim correctness. For fact-checking, however, we

*This work was supported by NSF grants CCF-15-13816, CCF-15-46392, IIS-14-08846, IIS-17-18398, IIS-18-14493, OIA-19-37143, an ARO grant W911NF-15-1-0408, a grant from the Knight Foundation, and a Google Faculty Award. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.

Proceedings of the VLDB Endowment, Vol. 12, No. 13
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3358701.3358708

must go beyond correctness. Instead, following the perturbation framework of [43, 44], we consider *perturbations* of the original claim, which provide a larger context in which we can assess various claim qualities—including *fairness*, *robustness*, and *uniqueness*—or to find counterarguments. The goal of data cleaning hence needs to be extended to help with such analyses.

An important question is how the objective of cleaning (i.e., how we define what "matter the most") affects fact-checking. A reasonable objective is minimizing uncertainty in some numeric measure of claim quality (e.g., fairness, uniqueness)—the goal is to ascertain claim quality. Another possibility is maximizing the chance of finding a counterargument to the given claim after cleaning—the goal is to purely counter the claim. One key question is whether these two goals align with each other. If not, fact-checkers need to be careful in avoiding potential bias of their data cleaning choices introduced by their desire to counter claims.

EXAMPLE 2. *To illustrate, consider the numbers of crimes in a particular jurisdiction in recent years (subscripts are years):*

X_{2014}	X_{2015}	X_{2016}	X_{2017}	X_{2018}
9,010	9,275	9,300	9,125	9,430

Suppose the data may contain errors. Cleaning each X_i may yield a number different from the above, but it would take considerable effort and we do not have enough resources to clean every X_i.

We wish to check the claim "crimes (in 2018) have gone up by more than 300 cases from last year," which attempts to put the blame on the current administration. This claim can be modeled as a simple query $X_{2018} - X_{2017}$. Obviously, cleaning X_{2018} and X_{2017} will let us tell whether the claim is outright incorrect.

But fact-checking goes beyond verifying correctness. Implicitly, this claim suggests that having an increase of 300 in a year is an unusual event. To assess whether this claim is really "unique," we consider a series of perturbations (more in Section 2.2), i.e., additional queries that help put the original one in context—how much did crimes go up by from 2016 to 2017, from 2015 to 2016, and from 2014 to 2015? We can then quantify the uniqueness of the original claim by counting the number of perturbations that yield a result no weaker than the original (i.e., > 300). To make this assessment, we would need a whole lot of data beyond the two specific years originally referenced.

The question of what data to clean now becomes more involved. Suppose our goal is to purely counter the claim by finding another instance of big increase in recent years. For this simple example, we would intuitively want to clean X_{2015}. If the result goes up just by a little, say, from 9,275 to 9,315, we will be able to make the counterargument that crimes went up just as much from 2014 to 2015 (implying that the previous administration could be blamed too). In contrast, cleaning X_{2016} is probably not a good investment, because it will unlikely yield a high enough number to make the increase over 2015–2016 significant, or a low enough number to make the increase over 2016–2017 significant.

On the other hand, the cleaning strategy above begs the question of whether it is "cherry-picking" in a way that can lead to unfair assessment of the original claim. A more impartial objective would be minimizing uncertainty in some measure of claim quality (e.g., uniqueness here). Would this objective lead to very different choices of data items to clean? Are there situations where two objectives actually align with each other? These are some of the questions this paper seeks to answer.

Note that this paper focuses on how to select data to clean, as opposed to specific data cleaning techniques. We assume that a cleaning procedure can resolve the uncertainty in a value by paying a cost; our algorithms are general enough to work for multiple scenarios, including manual cleaning.

Our contributions. Given a claim to check against a database with uncertain values, we solve the problem of choosing what values to clean under a cost budget in order to—roughly speaking—either 1) minimize uncertainty in some measure of claim quality, or 2) maximize the chance of finding a counterargument after cleaning. By appropriately defining a query function f over the database, we show that the above problem reduces to the following, more fundamental problems of choosing data to clean under a budget constraint, with different objectives:

- (MinVar) Minimize the uncertainty in the result of f, or more precisely, the expected variance in the result of f after cleaning.
- (MaxPr) Maximize the probability that the result of f after cleaning deviates significantly from its result before.

These optimization problems in general have applications beyond fact-checking. We give hardness results and greedy algorithms that work well in practice. Under certain assumptions, we can exploit properties of the data distribution and query function f to obtain efficient algorithms with good approximation guarantees.[1]

We apply our results to fact-checking and evaluate our algorithms using experiments. Fact-checking poses hard instances of the above problems (e.g., query functions can involve indicator and quadratic functions), but our results are general enough to apply. For the question regarding the differing goals of fact-checking, we show that in general, these two goals do not align. More interestingly, we also show that, under certain assumptions that may be reasonable in practice, these two goals can in fact align with each other. These results provide practical tools and guidelines that help fact-checkers clean data effectively while avoiding potential bias.

2. MODEL

We will first define our problems generally in terms of a query function f over uncertain data. The objective is to clean some data to either minimize the uncertainty in the result of f, or maximize the probability that the result of f after cleaning deviates significantly from the result before. Then, we show how to map concrete fact-checking tasks to special instances of these general problems, by defining appropriate f and choosing an objective.

2.1 Problem Definition

Let $\mathcal{O} = (o_1, \ldots, o_n)$ be a set of n objects. We assume that their identities are certain but their *values* are not. Each object o_i has a *current value* $u_i \in \mathbb{R}$, which may be incorrect; let $\mathbf{u} = (u_1, \ldots, u_n)^\intercal$. We model the *true value* of o_i as a random variable X_i with support V_i, and assume that we know the joint probability distribution of $\mathbf{X} = (X_1, \ldots, X_n)^\intercal$ with support $\mathbf{V}$. Cleaning object o_i costs c_i, and reveals its true value drawn from $\mathbf{X}$.

A *query function* f is a real-valued function of the object values. Informally, given f and a cost budget C, our problem is to choose a subset of objects $T \subseteq \mathcal{O}$ to clean in order to 1) minimize the uncertainty in $f(\mathbf{X})$, or 2) maximize the probability that cleaning T leads to a large deviation from $f(\mathbf{u})$. We describe the two objectives further below and formally define the problems.

Given $T = \{o_{i_1}, o_{i_2}, \ldots, o_{i_{|T|}}\} \subseteq \mathcal{O}$ where $1 \leq i_1 < i_2 < \cdots < i_{|T|} \leq n$, we denote $(X_{i_1}, X_{i_2}, \ldots, X_{i_{|T|}})^\intercal$ by $\mathbf{X}_T$ and its support by $\mathbf{V}_T$. In general, by subscripting an n-vector with

[1] We note that our solution to MinVar is also of technical interest because of its connection to the submodular function maximization problem with applications in sensor placement to reduce uncertainty [27, 28]. Despite apparent similarity, there is an intriguing dichotomy between the two problems that necessitates different approaches. We point out this dichotomy in the full version of this paper [38].

$T = \{o_{i_1}, \ldots, o_{i_{|T|}}\}$, we mean the $|T|$-vector consisting of just those elements at positions $i_1, \ldots, i_{|T|}$.

Minimizing uncertainty. The outcome of cleaning T is uncertain. While cleaning a particular o_i always removes uncertainty in X_i, doing so may, counterintuitively, increase uncertainty in $f(\mathbf{X})$, as illustrated by the following example.

EXAMPLE 3 (UNCERTAIN EFFECT OF CLEANING). *Consider a database of three objects with binary values X_1, X_2, and X_3, and query function $f(\mathbf{X}) = \mathbf{1}[X_1 + X_2 + X_3 < 3]$, an indicator function that returns 1 if the sum of values is less than 3, or 0 otherwise. Intuitively, the indicator condition may become harder or easier to satisfy depending on the outcome of cleaning an object value, say X_1. As a concrete example, suppose X_1, X_2, and X_3 are independent Bernoulli random variables with success probabilities $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{4}$, resp. Without cleaning, $f(\mathbf{X}) = 0$ with probability $\frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{4} = \frac{1}{24}$. If we clean X_1, there are two outcomes: If $X_1 = 0$: here we know the sum cannot exceed 2, so $f(\mathbf{X}) = 1$ for sure, and uncertainty is reduced. Otherwise, $X_1 = 1$: in this case, $f(\mathbf{X}) = 0$ with probability $\mathbf{Pr}[X_2 + X_3 \geq 2] = \frac{1}{3} \cdot \frac{1}{4} = \frac{1}{12}$, which is closer to a toss-up than the probability of $\frac{1}{24}$ for the case without any cleaning. In other words, uncertainty has increased.*

Hence, when choosing what to clean, we can only minimize uncertainty in the expected sense. To this end, consider $\mathbf{V}_T$, which forms the sample space containing all possible outcomes of cleaning T. The uncertainty in the query function result (due to remaining uncertainty in $\mathcal{O} \setminus T$) can be regarded as a random variable over $\mathbf{V}_T$. Our objective is then to minimize its expected value.

There are various measures of uncertainty in random variables (e.g. entropy). In this paper we consider variance, which is useful when the actual spread and magnitude of the numerical quantity matters (for example the number of crimes in Example 1). Formally, we define the optimization problem MinVar as follows:

$$\text{(MinVar)} \ \min_{T \subseteq \mathcal{O}} \sum_{\mathbf{v} \in \mathbf{V}_T} \mathbf{Pr}[\mathbf{X}_T = \mathbf{v}] \cdot \mathbf{Var}[f(\mathbf{X}) \mid \mathbf{X}_T = \mathbf{v}], \quad \text{subject to:} \sum_{o_i \in T} c_i \leq C. \tag{1}$$

Note that the above definition assumes the value domains to be discrete; the case when they are continuous can be defined analogously by integrating a probability density function.

Notice that so far, the current object values ($\mathbf{u}$) have no bearing on the problem definition. However, in the scenario to be discussed next, the current values play a more important role.

Maximizing surprise. The value of f before cleaning T is $f(\mathbf{u})$. By cleaning T, we replace, for each object $o_i \in T$, the current value u_i with a random draw of X_i, with the hope of lowering the value of f by more than a given threshold $\tau \geq 0$—intuitively, finding a "surprise." Because the outcomes are random, we maximize the probability that our goal is met. Formally, the optimization problem, which we call MaxPr, is as follows:

$$\text{(MaxPr)} \ \max_{T \subseteq \mathcal{O}} \mathbf{Pr}\left[f(\mathbf{X}) < f(\mathbf{u}) - \tau \mid \mathbf{X}_{\mathcal{O} \setminus T} = \mathbf{u}_{\mathcal{O} \setminus T} \right], \quad \text{subject to:} \sum_{o_i \in T} c_i \leq C. \tag{2}$$

Note the objective function has value 0 for $T = \varnothing$.

Discussion. Note that we assume each object o_i has a cleaning cost c_i; i.e., if we want to learn its correct value we need to pay c_i. This cost model has been widely used in data cleaning literature (e.g., [6, 7, 8, 31]). It is general enough to allow any individual costs to be specified, be they estimated or actual, monetary costs or human efforts. However, we do not yet consider more general cases where the cost function is non-additive.

We also assume that the distribution for each object's value is known. Such distributional knowledge can come in many ways, e.g., from the design of some sampling procedure, by modeling measurement errors of sensors, by resolving conflicting data from different sources [13], or by *opinion pooling* [5, 14, 26]. Such knowledge has been used in other related data cleaning models (e.g., [7]) as well as probabilistic databases (e.g., [4, 9, 10, 40]).

2.2 Application in Fact Checking

We now show how to apply the general problems defined earlier to concrete fact-checking tasks. Following [43, 44], we model a claim as a query (called the *claim function*) over a database instance. Let q° denote the claim function for the "original" claim to be checked, which returns $q^\circ(\mathbf{u})$ given the current values of the database objects. Intuitively, q° captures the original claim's particular view of the data. Checking this claim involves considering various *perturbations* to q° and see how they compare with $q^\circ(\mathbf{u})$. Let $Q = \{q_1, \ldots, q_m\}$ denote the set of m *perturbations*, each of which is a claim function obtained by changing (the parameters and/or form of) q° in some way.

A real-valued *relative strength function* $\Delta(\cdot, \cdot)$ is used to compare two claim function results: if $\Delta(q_k(\mathbf{u}), q^\circ(\mathbf{u}))$ is positive (negative), then q_k strengthens (weakens, resp.) q°; the absolute value of $\Delta(q_k(\mathbf{u}), q^\circ(\mathbf{u}))$ measures the extent of strengthening (weakening, resp.).

Not all perturbations are equally relevant to the original claim. For example, a perturbation q_k whose parameters are close to those of q° is more relevant than one whose parameters are far away. Therefore, we associate each perturbation q_k with a *sensibility* $s_k \geq 0$ such that $\sum_{1 \leq k \leq m} s_k = 1$. The higher the sensibility, the more relevant this perturbation is to q°. Together, the sensibilities $\mathbf{s} = (s_1, \ldots, s_m)$ define a probability distribution over Q.

EXAMPLE 4 (WINDOW AGGREGATE COMPARISON CLAIMS). *A* window aggregate comparison *[43] claim compares the aggregate values computed over two time windows of equal length. A real-life example is a claim made by Rudy Giuliani in 2007 [24], which stated that "adoptions went up 65 to 70 percent" in New York City between the periods 1990–1995 and 1996–2001. Here, the claim function is a linear function over subsets of values in the windows compared; i.e., $q^\circ(\mathbf{u}) = \sum_{i=l}^{l+w-1} u_i - \sum_{i=r}^{r+w-1} u_i$, for $1 \leq l, r \leq n$, where w is length of each window and u_i is the number of adoptions in year i. The first summation is over values in the earlier window, while the second summation is over the later window. The Δ function in this case is simply the difference between $q_k(\mathbf{u})$ (perturbation) and $q^\circ(\mathbf{u})$ (original claim). Sensibility of a perturbation q_k in this case may be defined to decay exponentially over its distance to q°, as measured by the number of years between the endpoints of their comparison periods. The intuition is that we care mostly about perturbations with temporal contexts similar to the original claim, which is when Guiliani was the mayor.*

Ascertaining claim quality. The following measures of claim quality were introduced in [43]. They all involve summarizing, in some way, over all perturbations, the difference between the result of each perturbation and that of the original claim function. In the following, $\mathbf{X} = \mathbf{u}$ if there is no uncertainty in the object values.

- ***Fairness*** can be measured by the amount of *bias* in $q^\circ(\mathbf{u})$, defined as $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X}) = \sum_{1 \leq k \leq m} s_k \cdot \Delta(q_k(\mathbf{X}), q^\circ(\mathbf{u}))$. Intuitively, bias of 0 means perturbations on average return the

same result as the original claim, so the original claim is fair. Negative bias means the original claim exaggerates, while positive bias means it understates.

- ***Uniqueness*** can be measured by the degree of *duplicity* in $q^\circ(\mathbf{u})$, defined as $\mathsf{dup}(q^\circ(\mathbf{u}), \mathbf{X}) = \sum_{1 \le k \le m} \mathbf{1}[\Delta(q_k(\mathbf{X}), q^\circ(\mathbf{u})) \ge 0]$. Intuitively, duplicity is the number of perturbations that yield stronger results than the original claim. The lower the duplicity, the more unique the claim.
- ***Robustness*** can be measured by the *fragility* in $q^\circ(\mathbf{u})$, defined as $\mathsf{frag}(q^\circ(\mathbf{u}), \mathbf{X}) = \sum_{1 \le k \le m} s_k \cdot (\min\{\Delta(q_k(\mathbf{X}), q^\circ(\mathbf{u})), 0\})^2$. Intuitively, low fragility implies that the original claim is robust; i.e., it is difficult to find perturbations that weaken the original claim.

When the object values $\mathbf{X}$ are uncertain, the results of claim functions are uncertain, so the claim quality measures become random variables over $\mathbf{X}$, whose uncertainty can be measured by their variance. A reasonable goal for a fact-checker, given a limited budget for data cleaning, would be to clean a subset of the object values in order to minimize the variance in some measure of claim quality. Because the outcome of data cleaning is uncertain, we minimize the expected variance over all possible outcomes. This problem is hence an instance of MinVar introduced in Section 2.1, with query function f set to the corresponding claim quality measure.

Increasing the chance of finding counterarguments. Consider a fact-checker with a "random" strategy, who picks a perturbation at random—with probability proportional to its sensibility—and hopes that it weakens the original claim q°. Given the current object values $\mathbf{u}$, $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{u})$, the bias in $q^\circ(\mathbf{u})$ as defined in Fairness, computes the expected extent to which the original claim will be weakened by a random perturbation. Intuitively, if the bias is well below some (negative) threshold, then we have a good chance of finding a strong counterargument to the original claim.

With data uncertainty, we can choose to clean a subset of the object values, and arrive at a new database instance $\mathbf{u}'$ consisting of the resulting values as well as old values from $\mathbf{u}$ for any objects not cleaned. The bias in $q^\circ(\mathbf{u})$ computed on this new database instance, $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{u}')$, would reflect how easy it is to find a strong counterargument after cleaning. Because the outcome of cleaning is uncertain, the amount of improvement—between $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{u}')$ and the baseline of $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{u})$—is uncertain. If our goal is purely to counter q°, we would like to choose objects to clean in a way to maximize the probability that the improvement is tangible, i.e., $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{u}') < \mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{u}) - \tau$, where $\tau \ge 0$ is a user-define threshold. This problem is hence an instance of MaxPr in Section 2.1 with query function $f(\mathbf{X}) = \mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X})$.

Comparing the two objectives. We now return to the interesting question raised at the beginning of this paper: how do the two objectives above—ascertaining claim quality and increasing the chance of finding counterarguments—compare with each other?

Under certain assumptions the two objectives do agree—namely, if $\mathbf{X}$ is a multivariate normal distribution centered at the current values $\mathbf{u}$ (independence assumption not needed), then for linear claim functions, maximizing the chance of finding a counterargument is equivalent to reducing the uncertainty in fairness. We formally state the result in the end of Section 3. Note that the assumption above is not unreasonable in practice. Oftentimes we have limited prior knowledge of the distribution of $\mathbf{X}$, and there is no reason to believe that database values are biased. In such scenarios, a multivariate normal with current values at the center would indeed be a reasonable starting assumption, and fact-checkers can rest assured that the goal of finding counters is equivalent to that of developing a better understanding of the fairness of the original claim.

However, the two objectives do not generally agree. Next, we show how they disagree on a simple concrete example involving only independently and uniformly distributed values and in Section 4, we empirically evaluate how much these objectives diverge.

EXAMPLE 5 (DIFFERING FACT-CHECKING OBJECTIVES). *We consider a database of two objects with values $\mathbf{X} = (X_1, X_2)^\intercal$, where X_1 and X_2 are independently and uniformly distributed over $\{0, \frac{1}{2}, 1, \frac{3}{2}, 2\}$ and $\{\frac{1}{3}, 1, \frac{5}{3}\}$, resp. The current (uncleaned) values are $\mathbf{u} = (1, 1)^\intercal$. Note that $\mathbf{Var}[X_1] = \frac{1}{2}$ and $\mathbf{Var}[X_2] = \frac{8}{27}$.*

The claim function to be checked is $q^\circ(\mathbf{X}) = X_1 + X_2$. Suppose the only relevant "perturbation" of q° is itself (i.e., $Q = \{q^\circ\}$), so $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X}) = X_1 + X_2 - q^\circ(\mathbf{u}) = X_1 + X_2 - 2$. We are given enough budget to clean either X_1 or X_2, but not both. We compare the objective of reducing the expected variance in $\mathsf{bias}(q^\circ, \mathbf{X})$ versus that of increasing the chance of finding a strong counterargument, where $X_1 + X_2 < \frac{17}{12}$ (below the baseline of $q^\circ(\mathbf{u}) = 2$).

For the first objective, note that with no cleaning at all, $\mathbf{Var}[\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X})] = \mathbf{Var}[X_1] + \mathbf{Var}[X_2] = \frac{1}{2} + \frac{8}{27}$; cleaning X_1 reduces it to $\frac{8}{27}$ while cleaning X_2 reduces it to $\frac{1}{2}$. Hence, the optimal choice (minimizing uncertainty) is to clean X_1.

For the second objective, if we clean X_1 (and leave $X_2 = 1$), we have $\mathbf{Pr}[X_1 + X_2 < \frac{17}{12}] = \mathbf{Pr}[X_1 < \frac{5}{12}] = \frac{1}{5}$; if we clean X_2 (and leave $X_1 = 1$), we have $\mathbf{Pr}[X_1 + X_2 < \frac{17}{12}] = \mathbf{Pr}[X_2 < \frac{5}{12}] = \frac{1}{3}$. Hence, the optimal choice is to clean X_2.

3. ALGORITHMS

The problems in Section 2.1 are difficult in general—even simpler forms are NP-hard (for example, the well-known Knapsack problem can be easily reduced to a special instance of the MinVar). We begin in Section 3.1 with simple greedy algorithms that prioritize objects to clean until the cost budget is exceeded. These algorithms are used as general heuristics for our problems.

We consider certain properties of the data distributions and/or query functions in Sections 3.2 and 3.3 that enable better theoretical bounds for greedy algorithms or more sophisticated algorithms. By assuming independent errors, we can show a number of interesting results: a) With an affine query function f, the objective functions of MinVar and MaxPr (additionally assuming zero-mean errors) become modularizable, allowing us to map these problems to Knapsack, for which better algorithms exist and even greedy offers constant-factor approximation. b) Remarkably, for any query function f, MinVar is equivalent to minimizing a submodular function with a cost upper bound constraint, allowing us to apply existing approximation algorithms with theoretical guarantees.

Finally, we return to the fact-checking applications in Section 3.4. We discuss how to compute expected variance (needed for running our algorithms) for fact-checking efficiently. We also show an interesting coincidence: under some conditions, namely when claims are linear and data errors are zero-mean multivariate normal, then the objective of minimizing uncertainty in claim quality (MinVar) and that of maximizing the chance of finding counters (MaxPr) in fact align with each other.

All formal proofs of the next lemmas and theorems can be found in the full version of the paper [38].

3.1 Greedy Algorithms

Algorithm 1 shows the template for our greedy algorithms. The code is parameterized by a *benefit estimation function* $\beta : \mathcal{O} \to \mathbb{R}$, to score objects for greedy selection. Intuitively, Greedy chooses the object with the highest benefit per unit cost ($\beta(o_i)/c_i$) to clean next until the total cost exceeds C. In the end, we ensure that we always find a good solution, in particular a 2-approximation, for a

Algorithm 1: Greedy. The algorithm is parameterized by $\beta : \mathcal{O} \to \mathbb{R}$, a function that returns the estimated benefit of cleaning a given object. In general, β may refer to the set T of objects that have already been chosen.

```
1 T ← ∅; c ← 0;
2 while ∃o_i ∈ O \ T : c + c_i ≤ C do
3     o_i ← arg max_{o_i ∈ O\T: c+c_i ≤ C} β(o_i)/c_i;
4     T ← T ∪ {o_i}; c ← c + c_i;
5 if O \ T ≠ ∅ then          // check to ensure 2-approximation
6     o_i ← arg max_{o_i ∈ O\T: c_i ≤ C} β(o_i)/c_i;
7     if β(o_i) > Σ_{o_j ∈ T} β(o_j) then
8         T ← {o_i};
9 return T;
```

class of objective functions [19] (special cases of our defined problems) by checking if the next object o_l with highest ratio $\beta(o_l)/c_l$ has larger benefit than the sum of benefits of the objects we have chosen before. For example, consider the well known 0-1 knapsack problem with two items x_1, x_2. The values of the items are $\beta(x_1) = 0.1$ and $\beta(x_2) = 10$, while the costs are $c_1 = 0.0001$ and $c_2 = 2$. With budget $C = 2$, the goal is to choose a set of items with cost at most 2 with the maximum value. Greedy would choose item x_1, because $\frac{0.1}{0.0001} > \frac{10}{2}$, and hence the value of the knapsack is 0.1. However, the optimal choice is to select item x_2 with value 10. In the end, our algorithm considers the benefit of the next non-cleaned object (Lines 5-8 in Algorithm 1) so it ensures that in such a case we take the item x_2 in our final solution.

A naive greedy algorithm. The simplest instance of Greedy, which we call GreedyNaive, uses the benefit estimation function $\beta(o_i) = \mathbf{Var}[X_i]$ (but 0 if the query function does not reference X_i). Note that $\beta(\cdot)$ does not depend on the objects already chosen, so one can sort $\mathcal{O}$ by $\beta(o_i)/c_i$ once and then proceed accordingly. Therefore, the running time of GreedyNaive is $O(n(t + \log n))$, where t is the complexity of computing each $\mathbf{Var}[X_i]$, which is $O(|V_i|)$.

While efficient, GreedyNaive tends to produce poor solutions in practice (see Section 4). Intuitively, GreedyNaive assumes that cleaning the value with the highest variance reaps the most benefit. At first glance, this estimation makes sense for both MinVar and MaxPr: the most uncertain object value may contribute the most to the query function result uncertainty, and a random draw of this value may cause the largest deviation from the original query function result. However, both assumptions easily fail in practice. Recognizing GreedyNaive's shortcoming of ignoring the objective when deciding what to clean, we next show a different greedy method that does consider the objective in its decision.

Estimating benefits from optimization objectives. A better strategy is to derive Greedy's benefit estimation function from the actual optimization objectives. We call the resulting greedy algorithms GreedyMinVar and GreedyMaxPr. Recall that T denotes the set of objects chosen so far. For $o_i \in \mathcal{O} \setminus T$, let δ_i denote the change in the objective if T changes to $T \cup \{o_i\}$. Let $\beta(o_i) = -\delta_i$ for GreedyMinVar and $\beta(o_i) = \delta_i$ for GreedyMaxPr.

Next, we show an example that illustrates why GreedyMinVar performs better than GreedyNaive even in the simple case where X_i's are mutually independent with unit cleaning cost, and the query function is symmetric in them.

EXAMPLE 6 (GreedyNaive VS GreedyMinVar). *Recall the database setup in Example 5, where X_1 and X_2 are independently and uniformly distributed over $\{0, \frac{1}{2}, 1, \frac{3}{2}, 2\}$ and $\{\frac{1}{3}, 1, \frac{5}{3}\}$, resp. Suppose objects have unit cleaning cost and we have budget to clean only one object.* GreedyNaive *will choose to clean X_1 because $\mathbf{Var}[X_1] > \mathbf{Var}[X_2]$.*

Consider MinVar *with query function $\mathbf{1}\left[X_1 + X_2 < \frac{11}{12}\right]$. Note that this function returns 1 for only two realizations of (X_1, X_2), namely $(0, \frac{1}{3})$ and $(\frac{1}{2}, \frac{1}{3})$. Thus, $\mathbf{Var}\left[\mathbf{1}\left[X_1+X_2<\frac{11}{12}\right]\right] = \frac{26}{225}$.*

GreedyMinVar *decides the item to clean by computing, for each item, the fraction of the variance improvement over the cost.*

If we clean X_1:

- *With probability $\frac{2}{5}$, $X_1 \in \{0, \frac{1}{2}\}$, so $\mathbf{1}\left[X_1 + X_2 < \frac{11}{12}\right] = 1$ with probability $\mathbf{Pr}\left[X_2 = \frac{1}{3}\right] = \frac{1}{3}$. Therefore, $\mathbf{Var}\left[\mathbf{1}\left[X_1+X_2<\frac{11}{12}\right] | X_1 \in \{0, \frac{1}{2}\}\right] = \frac{1}{3}(1 - \frac{1}{3}) = \frac{2}{9}$.*
- *With probability $\frac{3}{5}$, $X_1 \geq 1$, so $\mathbf{1}\left[X_1 + X_2 < \frac{11}{12}\right] = 0$ for certain. Therefore, $\mathbf{Var}\left[\mathbf{1}\left[X_1+X_2<\frac{11}{12}\right] | X_1 \geq 1\right] = 0$.*

Overall, the expected variance after cleaning X_1 is $\frac{2}{5} \cdot \frac{2}{9} = \frac{4}{45}$, and the improvement is $\frac{26}{225} - \frac{4}{45} \approx 0.0266$

On the other hand, if we clean X_2 instead:

- *With probability $\frac{1}{3}$, $X_2 = \frac{1}{3}$, so $\mathbf{1}\left[X_1 + X_2 < \frac{11}{12}\right] = 1$ with probability $\mathbf{Pr}\left[X_1 \in \{0, \frac{1}{2}\}\right] = \frac{2}{5}$. Therefore, $\mathbf{Var}\left[\mathbf{1}\left[X_1 + X_2 < \frac{11}{12}\right] \mid X_2 = \frac{1}{3}\right] = \frac{2}{5}(1 - \frac{2}{5}) = \frac{6}{25}$.*
- *With probability $\frac{2}{3}$, $X_2 \geq 1$, so $\mathbf{1}\left[X_1 + X_2 < \frac{11}{12}\right] = 0$ for certain. Therefore, $\mathbf{Var}\left[\mathbf{1}\left[X_1+X_2<\frac{11}{12}\right] | X_2 \geq 1\right] = 0$.*

Overall, the expected variance after cleaning X_2 is $\frac{1}{3} \cdot \frac{6}{25} = \frac{2}{25}$, and the improvement is $\frac{26}{225} - \frac{2}{25} = 0.0355$.

Hence, GreedyMinVar *chooses to clean X_2. The expected uncertainty after cleaning X_2 ($\frac{2}{25}$) is lower than that of cleaning X_1 ($\frac{4}{45}$). In other words,* GreedyMinVar*'s choice of cleaning X_2 is better than the choice of* GreedyNaive*'s choice.*

Since $\beta(\cdot)$ depends on T, we need to evaluate $\beta(\cdot)$ in every iteration. Therefore, a straightforward implementation of GreedyMinVar and GreedyMaxPr would have a time complexity of $O(n^2\gamma)$, where γ is the complexity of computing the objective function. This complexity is highly dependent on the forms of the data distribution and query function. Without any assumption about such forms, a brute-force implementation would enumerate all possible realizations of $\mathbf{X}$, implying that $\gamma = O(|\mathbf{V}|)$, which is exponential in n. To avoid this high complexity, one possibility is to estimate δ_i using Monte Carlo methods. Another possibility is to use more efficient algorithms for certain forms of data distributions and query functions; we defer that discussion to the next subsection.

3.2 Modular Objectives

We now examine some practical cases when we can prove good theoretical bounds for the greedy algorithms, or devise algorithms with even better guarantees. We start with a simple case where the optimization objective is essentially linear. An objective for MinVar or MaxPr is *modularizable over* $\mathcal{O}$ if it is equivalent to maximizing $\sum_{o_i \in T} w_i$ for some $\mathbf{w} = (w_1, w_2, \ldots, w_n) \in \mathbb{R}^n$, where $\mathbf{w}$ does not depend on T. Modular objectives have a number of applications in practise. For example, window aggregate comparison claims can be expressed as linear claim queries. In this case, notice that the fairness of a claim is a linear function, in the form $\mathbf{aX}$ where $\mathbf{a} = (a_1, a_2, \ldots, a_n)$. Hence, the problem of minimizing the variance of fairness of linear claims is a special case of optimizing a modular function over a weighted constraint.

The next lemma shows some conditions where MinVar, MaxPr problems have modular objectives.

LEMMA 3.1. *If components of $\mathbf{X}$ are pairwise uncorrelated and $f(\mathbf{X})$ is affine (i.e., $f(\mathbf{X}) = b + \mathbf{aX}$, where $\mathbf{a} = (a_1, a_2, \ldots, a_n)$),*

then MinVar *has a modularizable optimization objective, with* $w_i = a_i^2 \mathbf{Var}[X_i]$.

If components of $\mathbf{X}$ *are independently and normally distributed and centered around their current values (i.e.,* $X_i \sim N(u_i, \sigma_i^2)$*), and* $f(\mathbf{X})$ *is affine (i.e.,* $f(\mathbf{X}) = b + \mathbf{aX}$*, where* $\mathbf{a} = (a_1, a_2, \ldots, a_n)$*), then* MaxPr *has a modularizable optimization objective, with* $w_i = a_i^2 \sigma_i^2$.

It is easy to see that MinVar with modularizable objective is equivalent to the standard minimum knapsack problem: given a cost budget C', choose $T' \subseteq \mathcal{O}$ to minimize $\sum_{o_i \in T'} w_i$ subject to $\sum_{o_i \in T'} c_i \geq C'$. Similarly, the MaxPr problem with modularizable objective is equivalent to the standard maximum knapsack problem: given a cost budget C, choose $T \subseteq \mathcal{O}$ to maximize $\sum_{o_i \in T} w_i$ subject to $\sum_{o_i \in T} c_i \leq C$. This observation shows that MinVar and MaxPr are NP-hard problems. Using [2] we can get the following results.

LEMMA 3.2. *For* MinVar*, suppose the components of* $\mathbf{X}$ *are pairwise uncorrelated and* $f(\mathbf{X})$ *is affine. Let* t *denote the complexity of computing each* $\mathbf{Var}[X_i]$*. Then:*

- *An optimal solution can be computed in* $O(n(t + C))$ *time.*
- *For a parameter* $\epsilon > 0$*, an* $(1 + \epsilon)$*-approximate solution can be computed in* $O(nt + n^3/\epsilon)$ *time.*

Next we show the results for the MaxPr problem. We can take an exact pseudo-polynomial time algorithm for the maximum Knapsack problem and attain an exact solution for the MaxPr problem. Furthermore, an approximate solution for the equivalent Knapsack problem can be used to find an approximation solution to our original MaxPr problem.

LEMMA 3.3. *For* MaxPr*, suppose the components of* $\mathbf{X}$ *are independently and normally distributed and centered around their current values, and* $f(\mathbf{X})$ *is affine. Then:*

- *An optimal solution can be computed in* $O(nC)$ *time.*
- *Let* OPT *denote the optimal objective function value. There is an algorithm that returns in* $O(\frac{n^3}{\epsilon})$ *time a value* A *such that* $A = O(\mathrm{OPT})$ *if* $\mathrm{OPT} > 0.05$.

Greedy for modularizable objectives. By Lemma 3.1, the benefit estimation function for GreedyMinVar and GreedyMaxPr for the cases they cover, is simply $\beta(o_i) = a_i^2 \mathbf{Var}[X_i]$, the variance of X_i weighted by (the square of) its contribution to the query function—this estimation is in fact exact. Since the benefit does not depend on the objects already chosen, we can sort $\mathcal{O}$ upfront by $\beta(o_i)/c_i$, without computing $\arg\max$ in each iteration of the loop. Therefore, the running times of GreedyMinVar and GreedyMaxPr algorithms for these cases are the same as that of GreedyNaive, which is $O(n(t + \log n))$, where $t = O(|V_i|)$ for GreedyMinVar and $t = O(1)$ for GreedyMaxPr.

A well-known result on the knapsack problem is that Greedy achieves 2-approximation. Thus, by Lemma 3.1, GreedyMinVar provides a 2-approximation. Furthermore, by adapting the proof of Lemma 3.3 we can show that GreedyMaxPr provides a constant approximation for the MaxPr problem (similarly to the second part of Lemma 3.3).

3.3 General Query Functions

Now we consider the general case with arbitrary query function f. Interestingly, it turns out that as long as the X_i's are mutually independent, MinVar's objective function is *submodular* regardless of what the query function f is. A set function g is submodular if for any set $A \subset B$ and for any element $x \notin B$, $g(A \cup x) - g(A) \geq g(B \cup x) - g(B)$. This powerful observation enables us to draw techniques from the rich literature on submodular function optimization [15, 22, 35, 41]. These observations would lead to approximation algorithms for minimizing the expected variance of uniqueness, robustness or any other function over the claims. Notice that the results hold for any claim query (not only linear claim queries), as long as the random distributions X_i's are mutually independent.

Our main technical contributions are the following: If the random distributions X_i's are mutually independent then i) we show that the objective of MinVar is non-increasing and submodular regardless of what the query function is, and ii) we map MinVar to a minimization problem with a non-decreasing submodular objective function and a linear lower bound cost constraint. Finally, (ii) allows us to use the algorithms in [23] and get efficient approximation algorithms for MinVar.

In the following, let

$$\mathsf{EV}(T) = \sum_{\mathbf{v} \in \mathbf{V}_T} \mathbf{Pr}[\mathbf{X}_T = \mathbf{v}] \cdot \mathbf{Var}[f(\mathbf{X}) | \mathbf{X}_T = \mathbf{v}]$$

denote the objective function of MinVar.

We first start with another observation that $\mathsf{EV}(\cdot)$ is non-increasing, which holds in general, regardless of data distribution and query function. Then, we show that $\mathsf{EV}(\cdot)$ is submodular as long as the X_i's are mutually independent.

LEMMA 3.4. *The objective function of* MinVar *is monotone non-increasing in* T*; i.e., for all* $T \subseteq \mathcal{O}$ *and* $o' \in \mathcal{O}$, $\mathsf{EV}(T) \geq \mathsf{EV}(T \cup \{o'\})$.

LEMMA 3.5. *If components of* $\mathbf{X}$ *are mutually independent, then the objective function of* MinVar *is submodular in* T*; i.e., for all* $T \subset T' \subset \mathcal{O}$ *and* $o_j \in \mathcal{O} \setminus T'$, $\mathsf{EV}(T \cup \{o_j\}) - \mathsf{EV}(T) \geq \mathsf{EV}(T' \cup \{o_j\}) - \mathsf{EV}(T')$.

Therefore, MinVar with mutually independent X_i's is a problem of minimizing a non-increasing submodular function with a linear cost upper bound constraint. Next, we show that the MinVar problem can be mapped to a problem of minimizing a non-decreasing submodular function with a linear cost lower bound constraint. That will allow us to use known algorithms from the literature of submodular optimization. The key idea is that instead of choosing the subset T of objects to clean, we choose the subset $\overline{T}$ of objects to *not* clean (the cost constraint is complemented accordingly). Let $\overline{\text{MinVar}}$ be the problem defined as follows: Choose $\overline{T} \subseteq \mathcal{O}$ to minimize $\overline{\mathsf{EV}}(\overline{T}) = \mathsf{EV}(\mathcal{O} \setminus \overline{T})$ subject to $\sum_{o_i \in \overline{T}} c_i \geq \overline{C}$, where $\overline{C} = \left(\sum_{o_i \in \mathcal{O}} c_i\right) - C$.

LEMMA 3.6. *The* MinVar *problem can be mapped to* $\overline{\text{MinVar}}$*, with non-decreasing and submodular* $\overline{\mathsf{EV}}(\cdot)$.

Iyer and Bilmes in [23] propose efficient approximation algorithms for the problem of minimizing a non-decreasing submodular function with a submodular lower bound constraint. Notice that the $\overline{\text{MinVar}}$ problem has a linear lower bound constraint which is a (sub)modular function. Hence, we can use the algorithms in [23] to solve the $\overline{\text{MinVar}}$ problem and hence the MinVar problem (from Lemma 3.6). In particular, they present an algorithm with approximation ratio $O(\frac{H}{1-\kappa})$, where $\kappa = 1 - \min_{o_i \in \mathcal{O}} \frac{\mathsf{EV}(\varnothing) - \mathsf{EV}(\{o_i\})}{\mathsf{EV}(O \setminus \{o_i\})}$ (curvature of function $\mathsf{EV}(\cdot)$), and H, in our case, is the approximation ratio of an algorithm that minimizes a modular objective with a linear lower bound constraint (Knapsack problem). Using [2] we

can get in polynomial time a $O(1)$-approximation for the minimization knapsack problem, so that gives a $O(\frac{1}{1-\kappa})$-approximation for the MinVar problem. If $\kappa = 1$, we can use the observations in [23] to get a $O(\sqrt{n}\log n\sqrt{H}) = O(\sqrt{n}\log n)$-approximation algorithm for the MinVar problem.

The running time of the above algorithms is polynomial assuming that $\mathsf{EV}(\cdot)$ can be computed in polynomial time. However, in the worst case $\mathsf{EV}(\cdot)$ cannot be computed efficiently. There are instances of the $\mathsf{EV}(\cdot)$ function that are $\#P$-hard to compute exactly and no approximation algorithm is known [25]. In the next section we show how the function $\mathsf{EV}(\cdot)$ can be computed efficiently for fact checking applications.

Putting everything together we obtain the following result.

THEOREM 3.7. MinVar *has an* $O(\frac{1}{1-\kappa})$*-approximation algorithm that runs in polynomial time, assuming that the* $\mathsf{EV}(\cdot)$ *function can be computed in polynomial time, where* κ *is the curvature of* $\mathsf{EV}(\cdot)$*. If* $\kappa = 1$*, the approximation ratio is* $O(\sqrt{n}\log n)$*.*

If the objects have unit cleaning cost, applying the techniques from [18, 41], a *bi-criteria* approximation algorithm can be obtained. In particular, we can compute a set T such that $\mathsf{EV}(T) \leq \frac{1}{\alpha}\mathsf{EV}(T^*)$ and $\sum_{o_i \in T} c_i \leq \frac{1}{1-\alpha}C$, for any $0 < \alpha < 1$.

3.4 Application in Fact Checking

We now return to fact-checking and show how specific problems in this application domain can be solved using the algorithms proposed earlier in this section. We then return to the comparison between ascertaining claim quality and finding counters, and show when these two goals may align with each other.

General claims. As shown in Section 2.2, we formulate the problems of cleaning data to ascertain claim quality and to find counters as MinVar and MaxPr, respectively. Without any assumption on the data distribution or type of the claims, we can only solve the general instances of MinVar and MaxPr using GreedyMinVar and GreedyMaxPr, but without any theoretical guarantee on optimality. However, if we assume independent X_i's—in other words, errors in data values are independent—then by Theorem 3.7, we can apply the techniques from Iyer and Bilmes [23] to MinVar to obtain an efficient algorithm with approximation guarantees. In particular, for any claim, we show that we can compute the $\mathsf{EV}()$ function efficiently for fairness, uniqueness, and robustness, the three measures of claim quality introduced in Section 2 (and in [43]). This result implies that we can indeed run the algorithm in Theorem 3.7 for MinVar in polynomial time for fact-checking.

THEOREM 3.8. *Let* V *be the maximum support of distributions in* $\mathbf{X}$ *and* W *the maximum number of objects referenced by each claim. Assuming that the components of* $\mathbf{X}$ *are independent and* $q(\mathbf{u})$ *for each claim* $q \in Q$ *can be computed in* $O(W)$ *time, then for any set* $T \subseteq \mathcal{O}$*, the* $\mathsf{EV}(T)$ *of* $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X})$*,* $\mathsf{dup}(q^\circ(\mathbf{u}), \mathbf{X})$*, and* $\mathsf{frag}(q^\circ(\mathbf{u}), \mathbf{X})$ *can be computed in* $O(m^2V^{3W}W + n)$ *time.*

Note that in practice W is a small constant (e.g., $W = 8$ in Giuliani's claims), so we can compute the $\mathsf{EV}(\cdot)$ function and eventually run the algorithm from Theorem 3.7 in $O(\mathrm{poly}(n, m, V))$. Also note that if we always clean the values referenced by the original claim q° upfront, then the time to compute $\mathsf{EV}(\cdot)$ can be improved to $O(mLV^{2W}W)$, where L is the maximum degree of a claim (the degree of claim q is defined as the number of claims that share at least one object with q).

Linear claims. *Linear claim functions* are those that can be expressed in the form $\mathbf{aX}$ where $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ is a vector of weights associated with each object value. For example, the window aggregate comparison claims considered in Example 4 are linear: $a_i = -1$ if o_i belongs to the first window but not the second, 1 if o_i belongs to the second window but not the first, and 0 otherwise. In general, any SQL aggregation query over selections and joins is linear, provided that selection and join conditions involve only attribute values that are certain and therefore not included in $\mathbf{X}$.

We show that faster fact-checking algorithms can be developed for linear claims. Suppose the original claim function q° is linear; let $\mathbf{a}^\circ$ denotes the weights used by q°. Perturbations $q_1, \ldots, q_m$ of a linear query are linear too; let $\mathbf{A}$ be an $m \times n$ matrix where row $\mathbf{a}_{k,*}$ denotes the weights used by q_k; i.e., $q_k(\mathbf{X}) = a_{k,1}X_1 + a_{k,2}X_2 + \cdots + a_{k,n}X_n$. Let $\mathbf{a}_{*,i}$ denote the i-th column of $\mathbf{A}$, i.e., the weights used for X_i by the m perturbation queries. Further suppose that relative strength function $\Delta(\cdot, \cdot)$ simply subtracts its inputs, which is a natural choice for linear claim functions.

The fairness of the original claim can be ascertained by solving MinVar with query function $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X})$ (Section 2.2). This query function is linear for linear claim functions. More specifically, $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X}) = \mathbf{wX}$ where $w_i = \sum_{1 \leq k \leq m} s(q_k)(a_{k,i} - a_i^\circ)$. Hence, as long as the components of $\mathbf{X}$ are pairwise uncorrelated, the query function is modular and can be solved efficiently as a knapsack problem (Section 3.2).

Note that for the task of ascertaining claim qualities, linear claim functions do not always imply linear query functions for MinVar. For example, the query functions $\mathsf{dup}(q^\circ(\mathbf{u}), \mathbf{X})$, $\mathsf{frag}(q^\circ(\mathbf{u}), \mathbf{X})$ introduce non-linearity through with their additional use of indicator and quadratic functions. For MinVar with these two query functions, the results for modular objective functions do not apply. Instead, we can use Theorems 3.7 and 3.8 for constant W and find a good approximation for MinVar with $\mathsf{dup}(q^\circ(\mathbf{u}), \mathbf{X})$ or $\mathsf{frag}(q^\circ(\mathbf{u}), \mathbf{X})$ in polynomial time.

Now for the task of finding counters, we need to solve MaxPr with query function $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X})$ (Section 2.2). As discussed earlier, this query function is linear given linear claim functions. Therefore, we can solve this problem as a knapsack problem if the components of $\mathbf{X}$ are independently and normally distributed and centered around their current values (Section 3.2).

Ascertaining claim quality vs. finding counters. Finally, we return to the comparison between ascertaining claim quality and finding counters. As seen in Example 5 of Section 2.2, the two objectives in general differ. However, we show here that they turn out to agree with each other for linear claim queries. Following the analysis earlier in this subsection on linear claim queries, it is not hard to see that MinVar with query function $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X})$ (i.e., the goal of ascertaining claim fairness) and MaxPr with query function $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X})$ (i.e., the goal of finding counters) are aligned for linear claim functions when the components of $\mathbf{X}$ are independently and normally distributed and centered around their current values. In fact, we can extend this observation and show a stronger result for any multivariate normal distribution, without making the independence assumption.

THEOREM 3.9. *If* $\mathbf{X}$ *follows a multivariate normal distribution centered around the current values* $\mathbf{u}$*, and if all claim functions* $q^\circ, q_1, \ldots, q_m$ *are linear and the relative strength function* $\Delta(\cdot, \cdot)$ *is the subtraction operation, then for query function* $\mathsf{bias}(q^\circ(\mathbf{u}), \mathbf{X})$*, the optimal solutions to* MinVar *and* MaxPr *are the same.*

4. EXPERIMENTS

In this section, we experimentally evaluate the effectiveness of our proposed algorithms in achieving their objectives. We also

demonstrate their efficiency, and examine how different optimization objectives can lead to different outcomes, with implications to the practice of fact-checking. Our experiments use realistic claim functions and a combination of synthetic and real data. We use synthetic value uncertainty and cost distributions when such information is unavailable, and when we want to evaluate with different distributions. Details on the datasets are presented below; claim functions will be specified for individual experiments later.

Adoptions is a dataset derived from the number of adoptions in the New York City during 1989–2014. While the numbers were real, there was no published error model, so we assume that the data contains error modeled as follows. X_i, the number of adoptions in a particular year, follows a normal distribution with mean u_i, the current (reported) value, and standard deviation drawn uniformly from $[1, 50]$. We assume that the cost of cleaning each X_i is drawn uniformly at random from $[1, 100]$.

CDC-firearms and ***CDC-causes*** are real datasets complete with error models published by the Centers for Disease Control and Prevention (CDC) [1]. CDC routinely collects statistics on injuries and deaths by various causes through established sampling methods, and publishes the data along with statistics like standard errors, coefficients of variation, and 95% confidence intervals. Note that sampling procedures used by CDC ensure that the errors are independent and follow approximately normal distributions. For *CDC-firearms*, we get the estimated numbers of nonfatal firearms injuries in the USA during 2001–2017, along with the standard deviations. For *CDC-causes*, in addition to nonfatal firearms injuries, we also get the data on injuries due to transportation, as well as drownings and falls over the same time period; this results in a larger dataset with 68 values and enables more varieties of claim queries. We do not know the actual costs of cleaning, but a reasonable assumption is that acquiring older historical data is more expensive. Therefore, we generate cleaning costs in a way such that they decrease with recency: the cost of cleaning a value from the year 2001 is a random number in 195–200, the cost for 2002 is in 190–195, etc.

Synthetic datasets UR_x, LN_x, and SM_x are used to explore how different value distributions and dataset sizes affect various algorithms. For each value X_i, we first choose the size of its support uniformly at random from $[1, 6]$. Then, we generate the distribution for X_i with one of the following methods:

- UR_x generates fairly "random" distributions. We choose elements of $\mathrm{supp}(X_i)$ uniformly at random from $[1, 100]$ without replacement. For each element, we assign its probability in proportion to a number drawn uniformly at random from $(0, 1]$ (normalized such that the total over all of $\mathrm{supp}(X_i)$ is 1).
- LN_x generates skewed but unimodal value distributions. We start with a log-normal distribution with parameters $\mu = 0$ and σ chosen uniformly at random in $(0, 1]$. We quantilize distribution into as many equal-probability intervals as $|\mathrm{supp}(X_i)|$, and choose elements of $\mathrm{supp}(X_i)$ to be close to the right ends of these intervals. For each element, we then assign its probability in proportion to its probability density in the log-normal distribution (again, normalized to sum to 1). Note that resulting range is typically much smaller than the other two methods (which are based on $[1, 100]$).
- SM_x generates more complex multimodal distributions. We choose elements of $\mathrm{supp}(X_i)$ in the same way as UR_x, but for each element, we assign its probability in proportion to a random number in $(1, 0.1] \cup [0.9, 1]$, i.e., either low or high.

For cleaning cost, we draw it uniformly at random from $[1, 10]$ for each object. We have also experimented with a different cost distribution where individual costs are more extreme (either 1 or 10). It led to similar results and revealed no additional insights; hence we omitted the results here.

4.1 Effectiveness for Modular Objectives

We start by evaluating the effectiveness of our approach for the (simpler) case of modular objectives discussed in Section 3.2. We will focus on the scenario of minimizing uncertainty in the fairness measure of a window aggregate comparison claim (Example 4); our algorithm for this scenario is GreedyMinVar. (We omit the results for GreedyMaxPr because they are similar.)

Workloads. We consider Giuliani's adoption claim (Example 4) over *Adoptions*. The claim function subtracts the total adoption numbers over two four-year periods back-to-back (1993–1996 vs. 1989–1992). For analysis of fairness, we consider 18 perturbations of the original claim, all having the same form but each ending with a different year. To capture the intuition that important perturbations are those closest to the original in terms of the time periods compared, we let the sensibility of a perturbation decay exponentially (at rate $\lambda = 1.5$) over its distance to the original claim (as measured by the number of years between the endpoints of their comparison periods). Overall, the query function corresponding to fairness is complex albeit still linear, where the weight on each X_i incorporates the sensibilities of perturbations involving X_i.

We also consider claims over *CDC-firearms* and *CDC-causes*. For *CDC-firearms*, the claim function subtracts the numbers of firearms injuries over two four-year periods back-to-back, 2001–2004 vs. 2005–2008; such claims can be used to argue, for example, that current policy is more (or less) effective than the previous one. We consider 10 perturbations, with sensibility set up similarly as Giuliani's adoption claim. For *CDC-causes*, the claim function aggregates injury numbers across causes for comparison: "the number of injuries due to transportation is more than 30% of all other causes combined over the last 2-year period." We consider 16 perturbations, again with a similar sensibility setup.

Algorithms compared.

- Random simply chooses a next object to clean uniformly at random (with no replacement), until the budget is exceeded.
- GreedyNaiveCostBlind sorts the objects according to the variances in individual X_i's, and chooses them in descending order of uncertainty to clean, until the budget is exceeded.
- GreedyNaive (discussed in Section 3.1) estimates the benefit of cleaning X_i simply as its variance, and cleans objects in descending order of benefit/cost ratios.
- GreedyMinVar (our proposed algorithm) estimates the benefit of each X_i from the optimization objective—which considers the claim function as well as the structure and sensibilities of perturbations—and cleans objects in descending order of benefit/cost ratios.
- Optimum (discussed in Lemma 3.2) solves the same knapsack problem as GreedyMinVar, but uses dynamic programming to find the optimum solution in pseudo-polynomial time, which is much slower than GreedyMinVar.

Results and discussion. Figure 1 compares the effectiveness of the above algorithms in reducing uncertainty in claim fairness. The horizontal axis shows the cost budget as a percentage of the total cost of cleaning all data. Given a budget, we run each algorithm and plot the uncertainty (variance) that remains in the fairness of the original claim after cleaning the values chosen by the algorithm.[2] Note that the range of variance values depends on the dataset and

[2]Since Random randomly chooses data to clean, its result can vary significantly depending on its random choices. We conduct 100 runs of Random

claim in question, so we should focus on the relative performance of algorithms instead of absolute variance values.

From Figure 1a, we see a large gap between Random and other algorithms. With Random, uncertainty decreases linearly with increasing budget. Other algorithms are able to reduce far more uncertainty even with relatively low budgets; however, as budget increases, the reduction in uncertainty eventually slows down as expected. To better show the differences among other algorithms, we omit Random from other figures; Figure 1b is a zoomed version of Figure 1a. On all datasets, we see that our algorithm, GreedyMinVar, is very effective and almost indistinguishable from Optimum. For example, in Giuliani's adoption claim, with only 3% of the total cleaning cost, it is able to reduce uncertainty by a factor of 2.8. We also observe that GreedyMinVar clearly outperforms its less sophisticated cousins GreedyNaive (which ignores the query function of interest) and GreedyNaiveCostBlind (which further ignores differences in cleaning costs) in all datasets.

4.2 Effectiveness for Non-Modular Objectives

Here we focus on the scenario of minimizing uncertainty in claim uniqueness and robustness, where the optimization objective is no longer modular but the results of Section 3.3 apply.

Workloads. For minimizing the uncertainty on uniqueness, for *CDC-firearms* and *CDC-causes* datasets, we consider a claim of the form: "in the last two years, the number of injuries by firearms (or across four categories, resp.) is as low as Γ." To assess claim quality, we consider 8 perturbations, each summing 2 (or 8, resp.) object values. Intuitively, checking uniqueness involves counting how many perturbations yield results no higher than Γ. For minimizing the uncertainty on robustness, for *CDC-firearms* and *CDC-causes* datasets, we consider a claim of the form: "in the last two years, the number of injuries by firearms (or across four categories, resp.) is as high as Γ'." We consider the same number of perturbations as in the uniqueness case. Intuitively, checking robustness involves assessing how easy it is to find a perturbation that yields a result much lower than Γ'. Note that in all cases the query functions are non-linear. The goal is to choose a set of values to clean to minimize the variance in uniqueness or robustness.

We also use the three synthetic datasets in order to study how various value distributions affect the effectiveness of algorithms. The claim sums up 4 consecutive object value and states that the sum is as low as Γ (for uniqueness) or as high as Γ' (for robustness). For experiments on uniqueness, we generate small datasets with 40 uncertain values each, which make results easier to interpret (Section 4.4 experiments with larger datasets to evaluate efficiency); 10 perturbations of the original claim are used to assess uniqueness. For experiments on robustness, we generate bigger datasets with 100 uncertain values each; 25 perturbations are used to assess robustness. Note that the value of Γ or Γ' appearing in the original claim can affect both the initial uncertainty and how much reduction cleaning each value would bring, because certain sums can be more likely than others depending on the value distribution. To study this effect, we also test claims with different Γ/Γ' values.

Algorithms compared. Besides GreedyNaive and GreedyMinVar, we also implemented Best, the $O(\frac{1}{1-\kappa})$-approximation algorithm from [23] with theoretical guarantees described in Section 3. Note that Best does not guarantee an optimum solution, but it has the best known theoretical guarantee; hence, we use it as a yardstick for comparison. Since these algorithms work with discrete distributions, for CDC datasets, we discretize each normal distribution with the given mean and standard deviation using 6 and 4 discrete values for *CDC-firearms* and *CDC-causes*, respectively.

We do not compare our algorithms with those that focus on how to repair data instead of which data to repair; most algorithms surveyed in [16, 21] fall into this category. For example, [36] cleans up the data upfront and which data it cleans does not depend on the underlying objective function. These approaches may end up cleaning a lot of data that do not help with the underlying goal. This is particularly problematic to fact-checkers because in practice they are severely constrained by time and resources. The GreedyNaive algorithm we test in this section is in fact a representative instance of this approach and thus we have not implemented these algorithms.

Results and discussion. Figure 2 shows how various algorithms decrease uncertainty in claim uniqueness with increasing budget for *CDC-firearms* and *CDC-causes*. The horizontal axes show the budget, while the vertical axes show the (computed) expected variance after cleaning the values chosen by each algorithm. We observe that Best and GreedyMinVar have almost the same performance and they outperform GreedyNaive in all cases. For example, in Figure 2b, with 20% of the maximum budget, Best and GreedyMinVar find strategies that lead to half of the variance of what GreedyNaive is able to achieve. (Note that even though the range of variance values seems small, it needs to be interpreted in context—as the range of $\mathsf{dup}(\cdot,\cdot)$ is also small in this case, even a variance of 1 implies significant uncertainty.)

Figure 3 compares how various algorithms reduce uncertainty in claim uniqueness for the three synthetic datasets. Subfigures in the same row are for the same dataset, but each have a claim with a different Γ parameter. Overall, we find GreedyMinVar and Best effective across different value distributions, budgets, and claim parameters (Γ). Despite being simpler, GreedyMinVar is at least comparable to Best, and sometimes better. Generally speaking, they outperform the less sophisticated GreedyNaive, and the lead can be substantial. There is only one case where Best is slightly beaten by GreedyNaive (Figure 3b with $\Gamma = 200$ and enough budget), but even there GreedyMinVar consistently beats GreedyNaive.

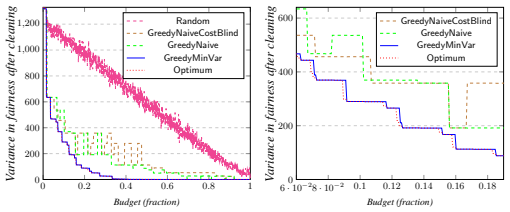
A second observation is the effect of Γ, which can be seen across the subfigures in each row of Figure 3. Generally, the initial uncertainty (when cleaning budget is 0, i.e., no data is cleaned yet) is the highest if Γ is in the midrange where the indicator functions in the query function could easily go either way—which confirms intuition. We discuss this observation more in the full version of this paper [38], where it becomes more visible with additional results.

As a related observation, when uncertainty is low to begin with, the advantages of GreedyMinVar and Best over GreedyNaive, in relative terms, are more pronounced. This observation is encouraging because in practice, most claims we are interested in involve Γ values that are out of ordinary, which correspond to regions where GreedyMinVar and Best significantly outperform GreedyNaive.

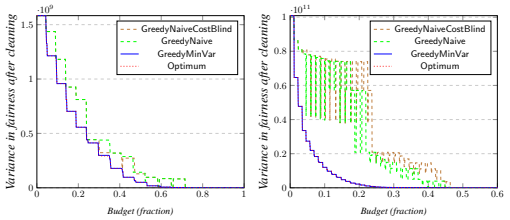
To sum up, GreedyMinVar consistently does the best job in removing uncertainty in uniqueness across various value distributions, budgets, and claims in these experiments. In some cases it even beats our yardstick, Best, but that should not be surprising since Best's guaranteed approximation factor depends on the curvature of the objective function (as discussed in Theorem 3.7), which means Best may not be optimum in practice.

Results on robustness are similar. Figure 4 samples some of the results, specifically for *CDC-firearms* and UR_x. Again, we see that GreedyMinVar and Best have almost the same performance and both outperform GreedyNaive. For example, for *CDC-firearms*, using the 30% of the budget, GreedyMinVar and Best reduce the expected variance to almost half of GreedyNaive. For UR_x, GreedyNaive performs much worse than GreedyMinVar and

and plot the average remaining variance. For other algorithms, the remaining uncertainty is computed exactly given their (deterministic) choices.

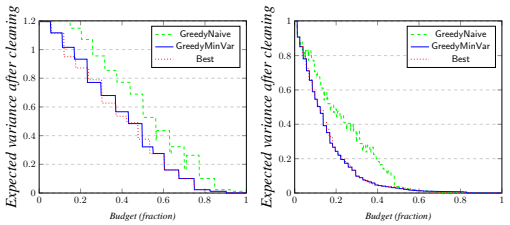


(a) *Adoptions* (b) Zoomed (a), no Random



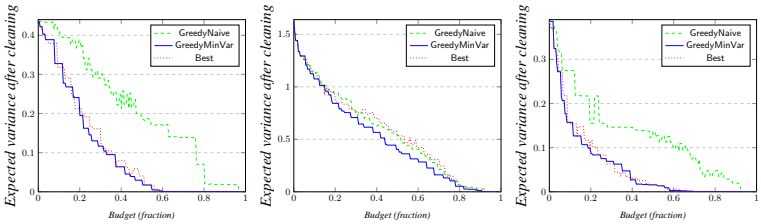
(c) *CDC-firearms* (d) *CDC-causes*

Figure 1: Effectiveness of algorithms in reducing uncertainty in claim fairness.

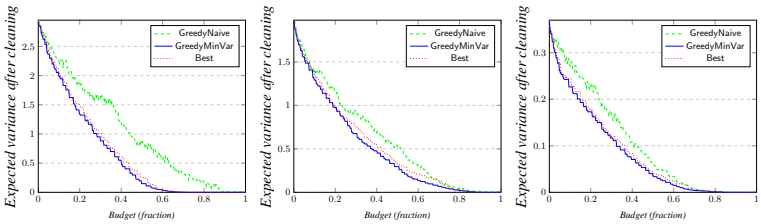


(a) *CDC-firearms* (b) *CDC-causes*

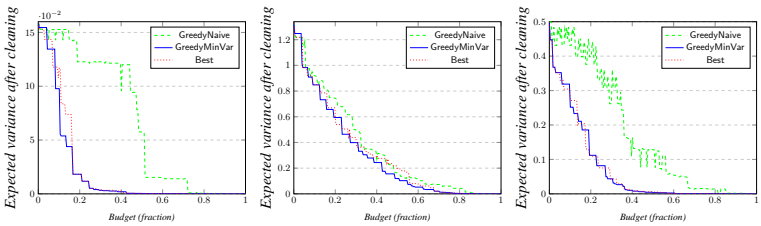
Figure 2: Effectiveness of algorithms in reducing uncertainty in claim uniqueness (CDC datasets).



(a) UR_x; claim with $\Gamma = 100$ (b) UR_x; claim with Γ=200 (c) UR_x; claim with $\Gamma = 300$



(d) LN_x; claim with $\Gamma = 3.5$ (e) LN_x; claim with $\Gamma = 4.5$ (f) LN_x; claim with $\Gamma = 5.5$



(g) SM_x; claim with Γ=100 (h) SM_x; claim with Γ=200 (i) SM_x; claim with $\Gamma = 300$

Figure 3: Effectiveness of algorithms in reducing uncertainty in claim uniqueness (synthetic datasets).

Best. Overall, the consistency of results on uniqueness and robustness is not surprising since our algorithms makes no assumption on the function used for ascertaining the claim quality (Section 3).

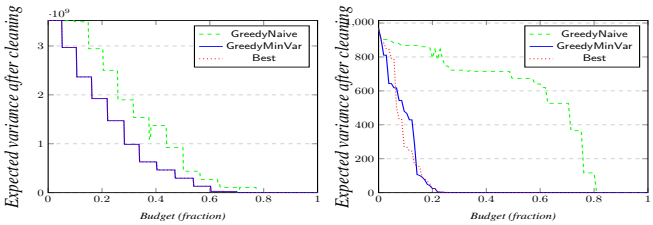
4.3 Effectiveness in Action

Ascertaining claim quality. Earlier in this section, we have seen how our proposed algorithms reduce the expected uncertainty in claim quality. However, given a specific scenario, a fact-checker using these algorithms to make data cleaning decisions will not necessarily experience the *expected* uncertainty—instead, after the true values of cleaned objects are revealed, a specific amount of uncertainty would remain. To help us evaluate the effectiveness of our algorithms in action from the perspective of a fact-checker, we perform experiments here to simulate a specific scenario. We consider the same data and workload as in Section 4.2. First, we establish the true values for all objects, which are generated from the given value distributions. These values are hidden from the fact-checker and our algorithms. Next, as the budget varies, we let each algorithm pick its set of objects to clean and reveal their true values, with which we can estimate the uniqueness of the claim (in terms of the mean and standard deviation of the degree of duplicity). Figure 5 plots the mean and standard deviation (respectively) of the estimates resulted from each algorithm's decision as functions of budget. The dataset here is *CDC-causes*, and the claim has the same form as in Figure 2b. For this specific scenario, the true degree of duplicity for the claim happens to be 5. From Figure 5, we see that Best and GreedyMinVar generate better estimates faster than GreedyNaive. For example, at 20% of the total cost, GreedyNaive's estimated mean is 3.7, with a standard deviation of 0.7, which is still difficult for the fact-checker to gauge true uniqueness. In comparison, GreedyMinVar and Best finds the mean to be 4.9 (which is closer to the true value) with a lower standard deviation of 0.4. While Figure 5 only illustrates one particular scenario, additional results in [38] support the same conclusion. Generally speaking, combining the results here and those in Section 4.2, we observe that GreedyMinVar in expectation requires cleaning less data than GreedyNaive for fact-checkers to assess claim qualities.

Finding counters. Similarly, we simulate scenarios to evaluate how our algorithms can help find counterarguments. We describe the results for one scenario on *CDC-firearms* here (other results in [38] are similar). To establish the (hidden) true values as well as the current (noisy) values, we randomly sample from the value distribution of each object. We want to check the claim that "in the past four year, we had only 310000 injuries by firearms, lowest in recent history." If we assume the current noisy values to be correct, there would be no counterexample in the database, i.e., there is no other period with fewer injuries. However, if we clean all data to reveal the true values, there is a counterargument for the period 2002–2006. A fact-checker must clean some tuples to counter the original claim. We observe that GreedyMaxPr uses only 7% of the budget to find the couterargument with high probability (more than 98%), while GreedyNaive uses 74% of the budget to achieve the same. The result here of course is for one specific scenario, but it does reaffirm the effectiveness of GreedyMaxPr in maximizing the probability of finding a counter.

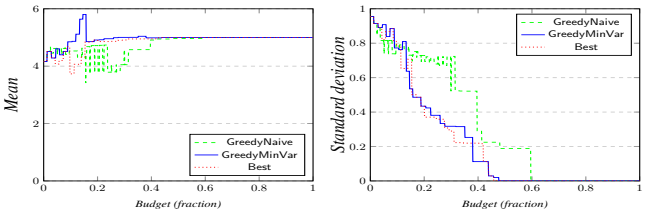
4.4 Efficiency

Having seen the effectiveness of GreedyMinVar in earlier experiments, we now evaluate its efficiency. We note that Best is gen-



(a) CDC-firearms (b) UR_x; claim with $\Gamma' = 100$

Figure 4: Effectiveness of algorithms in reducing uncertainty in the claim robustness (selected datasets).



(a) Mean (b) Standard deviation

Figure 5: Mean and standard deviation of estimates of claim uniqueness as functions of budget. *CDC-causes*.

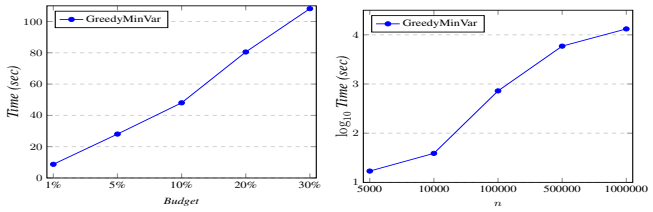
erally slower than GreedyMinVar (often by factors of more than 5 in our experiments) and does not seem to deliver better solutions in practice. On the other hand, GreedyNaive is much faster than GreedyMinVar because of naive benefit estimation, but it is not nearly as effective as shown earlier.

First, we consider the same scenarios as in Figure 3, but scale up each synthetic dataset to contain 10,000 uncertain values. We also proportionally increase the number of perturbations considered to 2,500 such that together they cover all values. We report the results for UR_x; other datasets are similar. Figure 6a shows the running time of GreedyMinVar as we give it increasing budgets to work with. We see that running time increases roughly linearly with budget. Even with a budget that allows 30% of all data to be cleaned, GreedyMinVar completes under 2 minutes.

Next, to study the effect of dataset size on running time, we consider progressively bigger datasets, from 50,000 to 1,000,000 uncertain values, whose distributions are still generated using UR_x. Again, we scale up the number of perturbations considered accordingly. We fix the budget at 5000 to allow about 1,000 values to be cleaned. Figure 6b shows how GreedyMinVar's running time (in $\log_{10}$ scale) increases with data size. We observe that each time that the data size increases by a factor of 10, the running time to clean about 1,000 tuples is 18–19 times larger. Even with a large dataset containing 100,000 uncertain values, it takes less than 12 minutes for GreedyMinVar to suggest cleaning 1,000 values, which translates to about 0.725 seconds per recommendation.

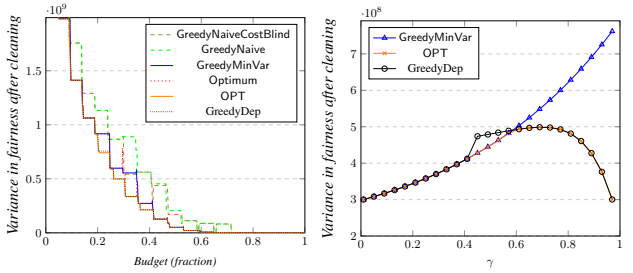
4.5 Handling Dependency

We note that our theoretical guarantees require the independence assumption among the object values. Nevertheless, our algorithms can still be applied to situations where the independence assumption does not hold. To see how our algorithms perform practically in such situations, we design additional experiments by modifying the *CDC-firearms* dataset. Recall that for each object X_i, CDC reports its standard deviation σ_i. Although errors across X_i's are actually independent because of CDC's data sampling procedure, we artificially introduce dependency as follows. We create a covariance matrix where the covariance between two objects X_i, X_j (where $i < j$ refer to the years) is given by $\gamma^{j-i}\sigma_i\sigma_j$, where the



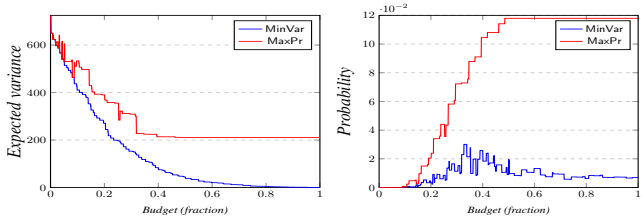
(a) $n = 10{,}000$, varying budget (b) varying n, budget $= 5000$

Figure 6: Running time of GreedyMinVar when reducing uncertainty in claim uniqueness. UR_x; claim with $\Gamma = 100$.



(a) $\gamma = 0.7$, varying budget (b) varying degrees of dependency; budget $= 30\%$

Figure 7: Effectiveness of algorithms in reducing uncertainty in claim fairness. *CDC-firearms* with dependencies injected.



(a) reducing uncertainty (objective of MinVar) (b) improving prob. of countering (objective of MaxPr)

Figure 8: How Optimum (for MinVar) and GreedyMaxPr (for MaxPr) achieve different objectives.

parameter $\gamma \in [0, 1]$ controls the degree of the dependency (the closer γ is to 1, the more dependent X_i's become). The exponent $j - i$ in this covariance model captures the intuition that the farther apart the two years are, the smaller their dependency is. The claim in question is the same as in Section 4.1.

Since we do not have an efficient algorithm capable of handling dependencies with good theoretical guarantees, for this experiment, where the dataset is thankfully small, we resort to a brute-force algorithm OPT and use it as a yardstick for comparison. OPT has full knowledge of data dependency (i.e., the covariance matrix), exhaustively considers all possible subsets of values to clean, and returns the best subset satisfying the cost constraint. Optimum, GreedyMinVar, GreedyNaive, and GreedyNaiveCostBlind are not made aware of any dependency at all. In addition, we implement a variant of GreedyMinVar called GreedyDep, which is given the dependency knowledge and uses it for estimating cleaning benefits.

Figure 7a compares how effective these algorithms are in reducing uncertainty in claim fairness, with parameter γ controlling the degree of dependency set to 0.7. We see that Optimum and GreedyMinVar always perform better than the simpler GreedyNaive and GreedyNaiveCostBlind. Furthermore, in many cases Optimum and GreedyMinVar have the same efficacy as OPT. For example, using only 18% budget, Optimum and GreedyMinVar have already reduced the uncertainty to less than half of the initial uncer-

tainty, matching the performance of OPT. Of course, as expected, having the knowledge of data dependency helps—there are cases where Optimum or GreedyMinVar still fail to match the performance of OPT. Interestingly, once given the knowledge of data dependency, GreedyDep, despite still being greedy, almost always matches the performance of OPT in this case. Overall, we observe that knowing the dependencies is beneficial, and the same greedy strategy is capable of reaping much of this benefit. Even if dependencies exist but are not known, GreedyMinVar and Optimum are still viable practical solutions for moderate degrees of dependency.

Figure 7b compares the effectiveness of algorithms when we vary the degree of data dependency, while fixing budget at 30%. If the dependency is weak enough, namely when $\gamma \leq 0.6$, then GreedyMinVar, even though it is not aware of any dependency, performs optimally. As dependency grows stronger, however, GreedyMinVar starts to fall behind OPT. Again, interestingly, GreedyDep almost always matches the performance of OPT, except for a small range of "middle" γ values, where intuitively the problem is the hardest (in contrast, having either independent values or highly correlated values makes it easier to resolve uncertainty). Overall, we conclude that even without any knowledge of dependency, GreedyMinVar (and Optimum) are viable as long as the degree of dependency is not too high. However, strong data dependencies would require an algorithm aware of such dependencies, but the greedy strategy is still effective.

4.6 Competing Objectives

Theorem 3.9 shows how the objectives of ascertaining claim fairness and increasing the chance of finding counters can be aligned if errors in data are normal and centered at 0. Here, we design experiments to show how the two objectives lead to very different outcomes when this assumption does not hold.

We return to the adoption scenario of Section 4.1, but simplify the claim to be about the sum over a 4-year window and consider perturbations with non-overlapping windows (same as in Section 4.2). Recall that Theorem 3.9 applies if all distributions are normal and centered around the current values. Here, we instead reassign the current values to random draws from these distributions, so they can deviate from the mean of the respective distributions.

For ascertaining fairness, we use Optimum as described in Section 4.1, which always finds the optimum solution. For maximizing the chance of finding counters, we use GreedyMaxPr. We run these two algorithms, but we also measure how they perform with respect to the objective that they are *not* optimizing for. Figure 8a compares how the two algorithms achieve the objective of ascertaining claim fairness, while Figure 8b compares how they achieve the objective of maximizing the chance of finding counters. The vertical axes show the respective objective function values, which are computed given the choices made by the algorithms.[3]

From Figure 8, as expected, each algorithm does well in terms of its intended objective. What is more revealing is how poorly they do with regard to the other objective. While Optimum quite effectively reduces the uncertainty in claim fairness (Figure 8a), it does not offer a good chance of finding counters even when given generous budgets (Figure 8b). On the other hand, GreedyMaxPr quickly increases the chance of finding counters (Figure 8b), but it is much less helpful in ascertaining claim fairness (Figure 8a). In fact, when given a budget above 48% of the total cost of cleaning all data, GreedyMaxPr simply refuses to clean any more values because doing so would actually decrease the chance of finding a counter (which explains why its achieved objective function values stay flat beyond this point). This questionable behavior illustrates the danger of a utilitarian approach of cleaning data for fact-checking that just seeks to maximize the chance of countering a claim.

[3]Note that the dataset's current values do not affect uncertainty in claim fairness, but do affect the chance of finding counters significantly. Hence, for Figure 8b, we repeat each experiment 100 times with different random draws of the current values, and report the average.

5. RELATED WORK

The fact-checking aspect of this paper builds on the framework in [43, 44]. However, that framework assumed an accurate database and did not consider data cleaning.

There is a rich body of literature on data cleaning; see [16, 21] for surveys. A number of papers have addressed the specific problem of cleaning data under a budget constraint, with the goal of improving query quality. Cheng et al. [7] proposed a metric for query result quality called *PWS-quality*, together with efficient algorithms for handling range and max queries. Mo et al. [33] further tackled top-k queries. The PWS-quality is based on entropy, and has nice properties for filter and ranking queries that, together with an independence assumption, result in modular optimization objectives. In contrast, our measure of uncertainty is based on expected variance; it is more suitable than entropy for numeric results, which arise naturally in the application of fact-checking. Our query functions are generally far more complex, which lead to non-modular, more difficult optimization problems. We also consider the alternative objective of maximizing surprise.

There are other models of data cleaning with different goals. For example, *ActiveClean* [29] proposed interactive data cleaning for statistical modeling. Given a budget, its goal was to choose a sample of data to clean while preserving provable convergence properties of stochastic gradient descent. *SampleClean* [42] used sampling to clean data in order to improve the quality of aggregate results by minimizing the impact of dirty data. It focuses more on improving estimation than on picking which value to clean. In [12] the authors propose *TARS*, which is a label cleaning advisor that provides valuable information when a model is trained or tested using noisy labels. While these approaches are also stochastic, their goals and technical challenges are quite different from ours. *HoloClean* [36] focuses on automatic repairing the whole dataset, given both constraints and known statistical properties of the input data. HoloClean is not directly applicable to our setting because our goal is not automatic repair of the whole dataset, but instead, selective cleaning of particular values to help fact-check a given claim.

There is also a large body of literature on approximation algorithms for stochastic data [11, 20, 30, 34, 39]. Particularly related is the line of work on sensing—how to place sensors or probe data in order to maximize utility under a budget constraint [27, 28]. In contrast to our setting, their actions have diminishing returns when taken later, and their direction of optimization is also different (minimization vs. maximization).

6. CONCLUSION

In this paper, we have considered how to help fact-checkers combat the issues of data quality and data fishing, by combining data cleaning and perturbation analysis, and by solving the optimization problem of choosing a subset of data to clean under a budget constraint, with the goal of either minimizing uncertainty in claim quality or maximizing the chance of find counters. We have demonstrated through experiments that our proposed algorithms are effective and efficient in practice. In sum, our results provide practical tools and guidelines that help fact-checkers clean data effectively while avoiding the potential bias introduced by their eagerness to counter claims.

7. REFERENCES

[1] Nonfatal injury reports. https://webappa.cdc.gov/sasweb/ncipc/nfirates.html. Accessed: 2019-07-15.

[2] C. Bentz and P. L. Bodic. A note on" approximation schemes for a subclass of subset selection problems", and a faster fptas for the minimum knapsack problem. *arXiv preprint arXiv:1607.07950*, 2016.

[3] C. Bialik. How to make sense of conflicting, confusing and misleading crime statistics. FiveThirtyEight, Jan. 2016.

[4] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *VLDB*, volume 87, pages 1–4, 1987.

[5] K. Chang. Combination of opinions: The expert problem and the group consensus problem. *Dissertation Abstracts International Part B: Science and Engineering[DISS. ABST. INT. PT. B- SCI. & ENG.],*, 47(3), 1986.

[6] J. Chen and R. Cheng. Quality-aware probing of uncertain data with resource constraints. In *International Conference on Scientific and Statistical Database Management*, pages 491–508. Springer, 2008.

[7] R. Cheng, J. Chen, and X. Xie. Cleaning uncertain data with quality guarantees. *PVLDB*, 1(1):722–735, 2008.

[8] M. Dallachiesa, A. Ebaid, A. Eldawy, A. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. Nadeef: a commodity data cleaning system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 541–552. ACM, 2013.

[9] N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: diamonds in the dirt. *Communications of the ACM*, 52(7):86–94, 2009.

[10] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *PVLDB*, 16(4):523–544, 2007.

[11] B. C. Dean, M. X. Goemans, and J. Vondrdk. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 208–217. IEEE, 2004.

[12] M. Dolatshah, M. Teoh, J. Wang, and J. Pei. Cleaning crowdsourced labels using oracles for statistical classification. *PVLDB*, 12(4):376–389, 2018.

[13] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *PVLDB*, 2(1):550–561, 2009.

[14] S. French. Updating of belief in the light of someone else's opinion. *Journal of the Royal Statistical Society. Series A (General)*, pages 43–48, 1980.

[15] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

[16] V. Ganti and A. D. Sarma. *Data Cleaning: A Practical Perspective*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013.

[17] A. Gathright. Darryl glenn: 'neighborhoods have become more violent' under obama's watch. PolitiFact Colorado, July 2016.

[18] A. Hayrapetyan, D. Kempe, M. Pál, and Z. Svitkina. Unbalanced graph cuts. In *European Symposium on Algorithms*, pages 191–202. Springer, 2005.

[19] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM)*, 22(4):463–468, 1975.

[20] T. Ilhan, S. M. Iravani, and M. S. Daskin. Technical note-the adaptive knapsack problem with stochastic rewards. *Operations research*, 59(1):242–248, 2011.

[21] I. F. Ilyas and X. Chu. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends in Databases*, 5(4):281–393, 2015.

[22] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

[23] R. K. Iyer and J. A. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pages 2436–2444, 2013.

[24] B. Jackson. Levitating numbers. FactCheck.org, May 2007.

[25] J. Kleinberg, Y. Rabani, and É. Tardos. Allocating bandwidth for bursty connections. *SIAM Journal on Computing*, 30(1):191–217, 2000.

[26] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[27] A. Krause, H. B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9(Dec):2761–2801, 2008.

[28] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.

[29] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. Activeclean: Interactive data cleaning for statistical modeling. *PVLDB*, 9(12):948–959, 2016.

[30] J. Li and Y. Liu. Approximation algorithms for stochastic combinatorial optimization problems. *Journal of the Operations Research Society of China*, 4(1):1–47, 2015.

[31] Z. Liu, K. C. Sia, and J. Cho. Cost-efficient processing of min/max queries over distributed sensors with uncertainty. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 634–641. ACM, 2005.

[32] M. D. Maltz. Bridging gaps in police crime data. U.S. Department of Justice, Office of Justice Programs, Bureau of Justice Statistics, sep 1999.

[33] L. Mo, R. Cheng, X. Li, D. W. Cheung, and X. S. Yang. Cleaning uncertain data for top-k queries. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 134–145. IEEE, 2013.

[34] D. P. Morton and R. K. Wood. On a stochastic knapsack problem and generalizations. In *Advances in computational and stochastic optimization, logic programming, and heuristic search*, pages 149–168. Springer, 1998.

[35] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[36] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.

[37] L. Robertson. Dueling claims on crime trend. FactCheck.org, July 2016.

[38] S. Sintos, P. Agarwal, and J. Yang. Data cleaning and fact checking: Minimizing uncertainty versus maximizing surprise. https://arxiv.org/abs/1909.05380.

[39] E. Steinberg and M. Parks. A preference order dynamic program for a knapsack problem with stochastic rewards.

Journal of the Operational Research Society, pages 141–147, 1979.

[40] D. Suciu, D. Olteanu, C. Ré, and C. Koch. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180, 2011.

[41] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM Journal on Computing*, 40(6):1715–1737, 2011.

[42] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 469–480. ACM, 2014.

[43] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu. Toward computational fact-checking. *PVLDB*, 7(7):589–600, 2014.

[44] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu. Computational fact checking through query perturbations. *ACM Transactions on Database Systems*, 42(1), 2017.