A New Generation of Tools for Scalable Web Accountability Featuring the Hubble Extensible Infrastructure

Paper #XX

Abstract

Privacy has all but disappeared from today's data-driven world. Users are eager to share their data online, and web services are eager to collect and monetize that data. As conventional confidentiality-based privacy becomes a distant dream, accountability and oversight become increasingly important to attain a balance between the data's commercial value and users' privacy and best interests. This paper puts forth *scalable web accountability*, a vision and research agenda whereby a new generation of web accountability tools – developed by the systems research community – are placed in the hands of privacy watchdogs, such as investigative journalists or the Federal Trade Commission, to let them monitor companies' use of personal data.

To facilitate the building of web accountability tools, we developed *Hubble*, the first scalable and extensible infrastructure that detects data use for targeting and personalization. Hubble's two main contributions include: (1) leveraging state-of-the-art statistical methods in unique ways to accurately detect targeting in black-box services based on experiments with differentiated user profiles, and (2) providing an extensible and dynamic architecture that assists web auditors throughout their examination process, from hypothesis specification to hypothesis exploration, validation, and refinement, all in real time and with solid statistical guarantees. We used Hubble to build two web accountability tools to pose specific questions about how Google targets ads within and across its services and how third-party web trackers use personal browsing histories to target ads on the web. We discovered solid evidence of Google's violation of two of its privacy statements, as well as multiple cases of third-party tracker targeting of children's websites.

1. Introduction

What do web trackers do with the data they collect about us, such as our browsing histories or mobile locations? Are our children's online activities specifically targeted by online advertisers? Do any shopping sites use our browsing profiles, or data from our Facebook accounts, to tailor their prices? Do any insurance sites do so? How does Google use our account data to target ads, news, or recommendations within and across its services? Does it use data within our accounts to target ads outside its services?

Myriad of questions exist about how web services use the information they collect about their users to target, personalize, or discriminate against them. Unfortunately, at present we have no good answers for these questions. At best, we may give abstract, small-scale, or dated answers (of the form "Staples was found in 2012 to tailor prices based on location"). Our goal is to enable much more concrete, current, and web-scale answers to such questions (of the form "Here is a list of the shopping sites on the Internet that tailor prices based on personal data as of March 25, 2015").

Indeed, the ability to monitor personal data use on the web is becoming increasingly important in today's world, in which the concept of personal privacy has come under siege. To defend themselves, those users aware of the risks they assume by entrusting private data to web services can choose from an array of end-user privacy tools, such as encryption, anonymity networks, location fuzzers, and ad/tracker blockers; however, many are unable to use such tools correctly or unwilling to use them at all if the tools sacrifice connectivity, convenience, functionality, or the ability to share data. Further, the tools themselves become rapidly obsolete as web services devise increasingly creative ways to track and leverage private data for their commercial ends.

While the conventional approach to protection provides tools for those end-users concerned about how their data is used, this paper argues for the urgent development and adoption of a new and complementary paradigm, one that places accountability and oversight at the foreground of the privacy arms race. *Scalable web accountability*, our vision and research agenda, aims to develop a new generation of web accountability and oversight tools for use by privacy watchdogs, be they investigative journalists, the Federal Trade Commission or other public agencies, or private entities that collect personal data and wish to themselves safeguard that

data. Our early discussions with a number of investigative journalists and FTC officers indicate great demand and significant potential for impact for such tools, which today are difficult to build due to a lack of scientific methods and infrastructures to enable them.

To facilitate the building of web accountability tools, we have developed *Hubble*, the first scalable, reliable, and extensible infrastructure system for answering questions about data use for targeting and personalization. The first paragraph of this section includes examples of the kinds of questions Hubble aims to support. To investigate a question of interest (e.g., "How do web trackers use the data they collect?"), a privacy watchdog (a.k.a., auditor) develops a set of targeting hypotheses (e.g., web trackers use the data to target ads on the web, or to personalize web page contents). The auditor then implements an API to specify these hypotheses in terms of putative personal data inputs (such as visited pages) that are thought to be used to target putative outputs (such as ads). Hubble then runs the experiments from the vantage points of multiple user profiles with differentiated inputs, and uses statistical correlation of the inputs and the outputs to predict the targeting. We find this methodology surprisingly flexible and supportive of many questions about targeting, personalization, and discrimination on the web.

Hubble's design brings two major research innovations. First, Hubble leverages state-of-the-art statistical methods to accurately detect targeting in black-box services. We show that these methods are well fit to the targeting problem, are accurate, precise, and flexible, and scale well with large numbers of hypotheses. Second, Hubble provides a dynamic architecture that assists web auditors throughout their examination process, including at-scale exploration of many potential hypotheses about targeting on the web, followed by real-time, self-driven validation and detailed investigation of those hypotheses that pan out; the result is statistically significant evidence for these hypotheses.

Atop Hubble, we have built two web transparency tools to investigate (1) how Google targets ads within and across its services and (2) how third-party web trackers use users' browsing histories to target ads on the web. Running relatively large-scale experiments with these tools, we demonstrate that Hubble is accurate, scalable, and flexible. Moreover, using these tools we found solid statistical evidence that contradicts two of Google's privacy statements related to ad targeting in their services. We believe that both of these stem from advertisers' abuses of Google's infrastructure. To address, we recommend improved filtering mechanisms, along with a more conservative formulation of the privacy statements. XXX trackes stuff?

Overall, our contributions are:

1. The first accurate targeting detection mechanism shown to work well at scale. It leverages state-of-the-art statistical methods with well-known scaling properties. In contrast, we show that prior work relying on in-house algorithms and scaling proofs scales poorly in practice [**?**].
2. The first infrastructure (Hubble) for web targeting experiments that assists web auditors throughout their examination process. It is scalable, robust, and extensible for both auditors and researchers.
3. A new, flexible API and experiment design methodology that can be used to answer at scale many questions about targeting on the web, including some questions asked by prior small-scale, limited targeting studies [**?**].
4. Two web transparency tools that answer interesting questions about ad targeting on the web.
5. We will release Hubble and its tools upon publication.

2. Motivation and Goals

Our goal is to commoditize the building of web accountability and oversight tools that monitor the use of personal data on the web. Our hypothesis is that a few core primitives can greatly facilitate the construction of a variety of tools to answer at scale many important questions about the data-driven web. This paper provides initial support for this hypothesis, which we call *scalable web accountability*. We test our hypothesis by designing and building *Hubble*, a scalable and extensible infrastructure that provides a first set of core primitives. We next use an example scenario to both motivate Hubble and derive requirements for its design.

2.1 Example Scenario

Ann, an investigative journalist, wishes to investigate how children and adolescents are targeted by various parties on the web, such as third-party web trackers and advertisers. She hypothesizes that advertisers might leverage information ammased by web trackers to bid for users with browsing histories characteristic of children or adolescents. Ann wishes to run a study to both quantify the amount of child-oriented targeting and find specific instances of what might be deemed as immoral or illegal targeting (e.g., targeting pronographic movies or recreational drugs at teenagers or promoting unhealthy eating habits to young children). Unfortunately, the number of websites dedicated to children is large (in the thousands according to Alexa), and there are even more neutral websites frequented by both children and adults on which ads targeted at children might appear.

Thus, Ann would like to run a *large-scale survey experiment* that tries out a large number of children's websites, looks for ads on even more websites, and finds those ads that target children's websites in particular (likely a small fraction of all ads she collects). For any case of illegal or immoral targeting, Ann plans to investigate through journalistic means (e.g., interview the advertiser, tracker, or other entity potentially responsible for the targeting) whether the targeting was intentional, a mistake, or the result of algorithmic choices. Such investigations are expensive, so Ann requires *high confidence* in a result to investigate it.

Ann involves her technical staff, Bob, in her experiment. Bob is a decent programmer with minimal background in statistics, but certainly not a distributed systems or statistics expert. Discussing the project, Ann and Bob quickly realize that running such an experiment at scale would be very challenging. Their initial idea is to take each pair of sites – one children's site (W_c) and some other site, W_o – and execute two sub-experiments for each pair (W_c, W_o): in one, they launch a browser and visit first W_c and then W_o to collect the ads; in the other, they launch a browser and visit only W_o. If they find a particular ad consistently appear in the first experiment but not in the second, they can deem it as targeted against W_c. But how should they define "consistently appear" so as to be confident of the prediction? And what are they missing by only testing two pairs of websites at a time? Maybe advertisers target ads only on profiles with sufficient children's websites in their histories. How could they possibly make their experiment scale to the tens of thousands of websites they wish to try all combinations? And how can they obtain high confidence in their results with all the noise that arises from visiting many different websites from one profile, on which many different trackers independently observe all visits and target ads based on different aspects in that profile? The study seems unamenable.

Enter Hubble. Bob decides to build their experiment on Hubble, which provides all the primitives needed to run his experiment at scale and with sound statistical guarantees. First, Hubble provides a *survey experiment primitive*, which lets an auditor efficiently explore a wide range of potential targeting hypotheses (such as childrens' website W_c is targeted by ads on website W_o); it combines those hypotheses to use minimal resources and relies on state-of-the-art statistical methods to weed out the noise and provide statistically significant correlations. Second, Hubble associates a well-defined *statistical confidence metric* with each targeting result so an auditor can interpret the significance of his results. Third, Hubble provides a *validation experiment primitive*, which automatically triggers a new, laser-focused experiment to obtain further evidence for results from survey experiments whose confidence is below acceptable level for an auditor (e.g., increase from 0.9% to 0.99% confidence).

To build their experiment on Hubble, Bob would have to write code to combine these primitives into an end-to-end experiment design. Hubble will run the experiment, analyze the data, validate the results, and output a set of ads that, with high confidence, target children's sites. However, in this case, we have already developed precisely this tool: it is called *TrackTheTracker* and supports investigations of targeting on browser histories.

2.2 Design Goals

The preceding example leverages Hubble to investigate a particular question, or hypothesis. Our primary goal in Hubble is to provide a flexible API and extensible architecture to support a wide class of questions pertinent to targeting, personalization, and discrimination. More specifically, Hubble aims to support questions that can be formulated as a set of hypotheses of the form *"Personal data input X is being used to target outputs of a particular kind."* We believe that all the questions listed in the first paragraph of §1 can be investigated through such hypotheses.

More formally, our goals are: **[xxx dirty]**

• *Extensibility:* Hubble must be extensible in two dimensions. First, it lets privacy watchdogs extend it to implement the tools necessary for their investigations. Second, it lets researchers develop new core primitives to support use cases that Hubble cannot currently support. For example, some auditors may require *causal assurances* and not only *high-confidence correlations*, as Hubble currently provides. We discuss potential routes for a causal inference primitive in §**??**.

• *Statistical justification for its inferences.* Hubble must provide statistical justification for its correlations. Any validations necessary to provide such guarantees must be run in real time, right after Hubble detects the correlation to ensure that the evidence.

• *Support technical but non-expert auditors.* Hubble's direct users must have basic understanding of statistics to interpret our confidence levels and understand the difference between correlation and causation.

• *Use established algorithms to detect targeting.* An important decision we made in Hubble was to leverage well-established statistical methods XXX.

We know of no prior system that achieves these goals. The closest contenders – XRay [19] and AdFisher [**?**], which aim to support targeting and personalization questions – are rather singular building blocks than real infrastructures designed to provide a comprehensive set of primitives to web auditors. Moreover, we find that these systems do not scale well in practice. XRay additionally provides no statistical guarantees for its results.

3. The Hubble Design

Hubble is a scalable, extensible, and robust infrastructure system for data targeting experiments. Developers and auditors use Hubble to build the tools necesary to answer high-level questions about personal data targeting on the web. §2 already gave examples of such questions. This section uses the following question for illustration: "How do web trackers use the data they collect to target ads?"

Briefly, to explore targeting questions with Hubble, a developer (1) designs a series of *experiments* that test, refine, and validate *targeting hypotheses* about how specific inputs are used to target web service outputs. (2) implements *the Hubble interface* to model those experiments based on well-defined rules that ensure Hubble's effectiveness, and (3) launches the experiments with Hubble, which uses user profiles with differentiated inputs to collect outputs and analyzes the results to identify any statistically significant evi-

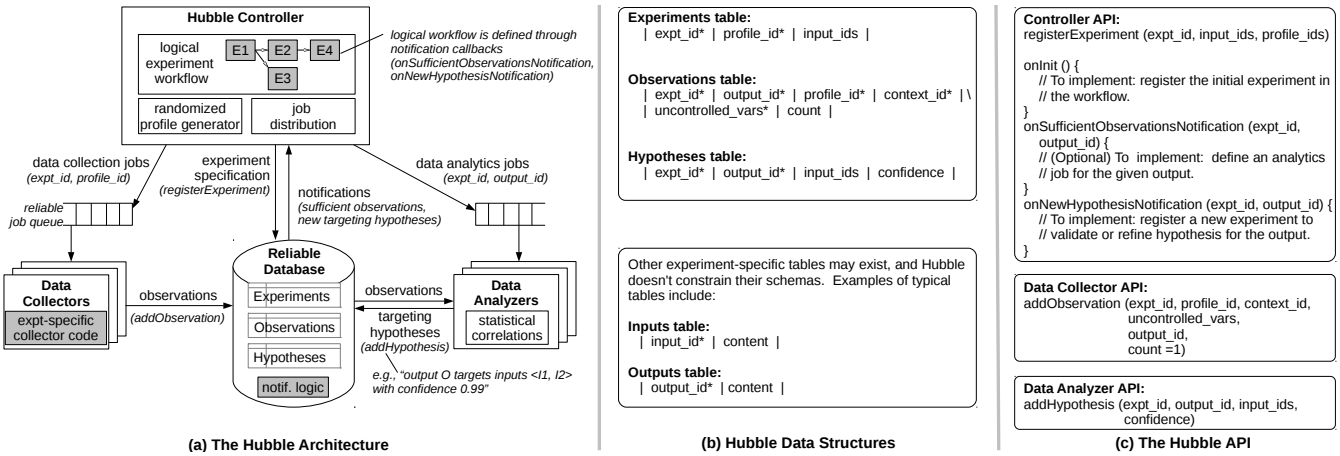

Fig. 1: **The Hubble Design.** (a) shows the Hubble architecture; grey boxes are experiment-specific, white boxes are Hubble components. Hubble has three main components: (1) the *Controller*, which coordinates the experiments and their analyses; (2) *data collection workers*, which run experiments and collect the data; and (3) *data analysis workers*, which analyze data from the experiments to detect targeting. These components are stateless; all state is maintained in a reliable, scalable database (DB). (b) shows the schema of Hubble's persistent state; * identifies the field(s) that form the primary key in each table. (c) shows the API that a developer implements and/or uses to build a data observatory tool on Hubble.

dence for the targeting hypotheses. Fig.5 illustrates Hubble's architecture and developer APIs.

3.1 Architecture

Hubble's architecture (Fig.5(a)) consists of three core components: (1) *data collection workers* that gather data outputs such as ads and recommendations from the web based on the user specified or Hubble generated profiles; (2) *data analytics workers* process the collected outputs to identify targeting through correlation; (3) the *Controller* handles notifications of new observations or hypotheses and starts new data collection or analytics workers as needed. All components are stateless and the state is persisted to a scalable, reliable database (DB). The controller can also generate profiles from sets of inputs. Hubble create and manages three tables that developers should not modify: `Experiments` maintains Hubble metadata about the user-defined experiments; `Observations` amasses the data obtained by the data collectors for use by the data collectors; `Hypotheses` contains all of the targeting hypotheses that Hubble validated.

In the architecture figure, white boxes denote tool-agnostic Hubble components; grey boxes denote tool-specific components that a developer implements. The most valuable generic component is the statistical correlation engine that identifies input/output targeting with minimal assumptions. The most coding-intensive component is the experiment specific data collection code. The data collection module emulates user inputs such as web browsing to create differentiated profiles and then collects outputs in the form of ads, recommendations, or other personalized facet; this typically involves a web crawler or browser automation, a conceptually simple but fairly coding-heavy process. The most intellectually challenging tool-specific component is the experiment design.

In the simplest form, an experiment is specified as a set of *inputs* that the developer hypothesizes might be used for targeting (e.g., the websites in a user's history might be used to target ads on the web). In practice, experiment designs are often more complex. They are specified as workflows of simple experiments that will survey an ecosystem, then validate and refine targeting hyptoheses. For example, a developer may create an initial experiment that collects information about a large number of sites that are suspected of having targeted ads and then a series of refinement experiments that determin what sites and which specific trackers ads target. Hubble's streamlined architecture supports this kind of experiment chaining and ensures collection in real time of sufficient evidence to both validate and refine targeting hypotheses.

To launch experiments in Hubble, a developer registers the first experiment in her workflow with the Controller by calling `registerExperiment` in the Hubble API. She specifies a unique ID for that experiment, the set of inputs on which to identify targeting (e.g., the set of webpages to visit), a set of profiles to exercise the inputs, and the data collection procedure to invoke for that experiment. Profiles can be either soft profiles (represented by cookies and other browser state) or accounts (such as Google accounts); soft profiles need no a priory set-up but accounts do. The Controller then assigns the inputs randomly to the different profiles and creates a data collection job for each profile. The jobs are distributed to data collection workers through a reliable job queue. One profile will be exercised by one data collection worker, which first populates its profile with the specified inputs (e.g., visits those pages) and then collects the service outputs offered to its profile (e.g., the ads shown on the visited pages). Whenever a data collector observes an output, it will report it to Hubble by calling the `addObservation`

function in the Hubble API. The function persists information about the context of the observation (such as which profile and others) into the `Observations` table in the reliable database for later analysis.

As motivated in §2, timeliness is vital for effective investigations of the ever-changing web. A key feature in Hubble is to both identify plausible targeting hypotheses and validate and refine them in as close to real time as possible. To this end, Hubble monitors the `Observations` table using a trigger-like mechanism installed in the DB. When sufficient data is available for a particular output O (e.g., when an ad was observed in the context of sufficient differentiated profiles), the DB triggers a notification to the Controller, which launches a data analytics job for that particular output O in an attempt to determine the inputs that it is targeting. The analytics job is picked up by an analytics worker, which leverages known statistical methods in unique ways to identify whether any subset of the inputs strongly correlate with the output, and if so which. In addition, the statistical methods also yield a metric of *confidence* that measures the statistical significance of their guess. All data needed to do the correlation is in the `Observations` and `Experiment` tables.

For example, using the information about the profiles in which ads were seen, statistical correlation may find that an ad O is often seen in profiles that include websites $I1$ and/or $I2$ in their histories, and never in profiles missing one or both of these websites. In such a situation, statistical correlation will conclude that O targets $\{I1, I2\}$ with high confidence (e.g., .99). This association, along with its confidence, will be added to the `Hypotheses` table in the DB. Other outcomes exist for the analytics job. First, the statistics may find that there is sufficient evidence in the observations to flag the ad as *untargeted* against any of the explanatory inputs, in which case the correlation engine will add O, $\{\}$, $confidence$ into the `Hypotheses` table. Note that the ad could still be targeted against aspects that the experiment did not model as inputs to vary in the experiments (e.g., the ad is targeting the city in which the data collectors were run). Second, the statistics may find that the evidence to make a targeting conclusion either way may be insufficient. In such situations, no targeting hypothesis is added to the database. If later on, more observations of the ad are amassed through data collection, then the correlation job will run again, which may enable a more precise outcome.

Like the `Observations` table, the `Hypotheses` table also has a trigger installed, which notifies the Controller whenever a new targeting hypothesis with some minimal confidence is added to it. Upon receiving the notification for a new targeting hypothesis (O, $\{I1, I2\}$, $confidence$), the Controller invokes a developer-provided callback, `onNewHypothesisNotification`, which determines the next steps. This is where a developer can register any validation and/or refinement experiments in her

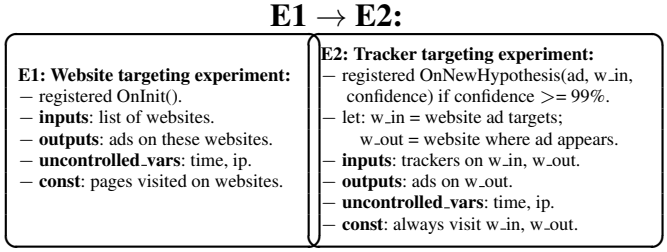

Fig. 2: **TTT Experiment Design.** E1 is a large-scale exploratory experiment to find cross-website targeting of ads. E2 is a focused refinement experiment to find which trackers target ads between particular pairs of websites.

workflow, which focuses on the newly discovered targeting hypothesis and either gathers more data to further increase the confidence or asks a different question (e.g., which specific tracker was responsible for targeting ads against webmd.com). To register a new experiment, the developer will use again the `registerExperiment` method in the Hubble API, and Hubble will launch that new experiment (or experiments if there are multiple) similarly to the starting experiment.

3.2 API

The Hubble API is best explained with an example. Fig.2 shows a simplified version of the experiment design in TTT, which we will use as an example throghout the rest of this section. The design has two phases: (1) a first exploration phase, which discovers cross-website targeting and (2) a second refinement phase, which for each targeting hypothesis for an ad A, it determines which tracker(s) were responsible for the targeting. In the simplest case (shown in the figure), each phase runs one experiment, chained one after the other: $E1 \rightarrow$ E2. In reality, a more complex design is needed for this experiment, which involves further experiments and is described in §3.4.

[xxx Naming scheme doesn't match API.] The Hubble API requires the developer to implement 4 experiment specific components along with an optional 5th component. First, the developer must populate the experiment `Experiment` with 3 callbacks: `init`, `onSufficientObservatio` and `onNewHypothesisNotification` . The `init` funtion is responsible for initializing the experiment including, populating the database with all inputs (pages to visit or search queries), creating all profiles that will be used to collect data, and optionally assigning inputs to profiles. By default, Hubble will assign each input to a profile with probability 0.5 but this can be overriden if an experiment requires more complex assignment. `onSufficientObservationNotificat` contains the functionality to respond to new observations. At it's simplest, `onSufficientObservationNotiviation` will create a hyptothesis job the `Experiment` but may implement more complex funcionality such as updating experiment specific analytics. `onNewHypothesisNotification`

is called the analytics worker generates new hypotheses with sufficient confidence. It's main responsibility is to start new experiments in the workflow if needed and publish results of the current experiment.

The developer must also implement the `startCollection` function on the `CollectionWorker` object. `startCollection` implements the functionality to collect all data for a specific profile and calling `addObservation` function for each output observed. For example, in *TrackTheTrackers* `startCollection` will drive a web browser to visit all of the web pages associated with a browsing profile and collected all of the trackers and display ads observed on those pages.

Last, the developer may override the `startAnalysis` function on the Hubble`AnalysisWorker`. Hubble provides robust statistical methods to detect targeting but these methods may not be appropriate for all circumstances. For these cases, Hubble allows the developer to implement their own analysis mechanisms.

[xxx Need to explain uncontrolled variables and other API. The section is too oriented toward how the system works. This is an API section, which needs to focus on the API (what not how).]

3.3 Statistical Correlation

A core contribution in this paper is to show how out-of-the-box, traditional statistical tools can be combined to accurately detect targeting and personalization in black-box services. Previous work aiming to discover targeting invented new algorithms and proved properties about them [**? ?**]; in our experience, these algorithms are fragile, inflexible, and do not scale with large numbers of hypotheses (contrary to claims). For Hubble, we opted to leverage established statistical methods with well understood properties and solid implementations to detect targeting; we find our mechanisms precise, scalable, and flexible. Moreover, we foresee great potential to tap into the enormous and highly active body of work on statistical correlation methods to develop increasingly robust and expressive targeting detection mechanisms.

This section describes how we combine known statistics to detect targeting from experiments with differentiated-input profiles.[1]

Linear and Logistic Regression. The core statistical method we build upon in Hubble to generate targeting hypotheses is *linear regression*. The basic linear regression model posits that an *output variable* y is determined by a linear combination of p *input variables* $x_1, x_2, \ldots, x_p$, plus a random noise term ε with mean zero. For example, the input variable x_7 could be a $\{0, 1\}$ indicator for whether a particular profile visits website W_7; the output could be the number of times the ad is observed in that profile. Using vector notations $\mathbf{x} := (x_1, x_2, \ldots, x_p)$ and $\mathbf{w} := (w_1, w_2, \ldots, w_p)$, the linear model is written as $y = \langle \mathbf{x}, \mathbf{w} \rangle + \varepsilon$. Here, $\mathbf{w}$ are the *regression coefficients*, and $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_i a_i b_i$ is the dot product between vectors.

Given n vectors of input values $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ together with their corresponding output values $y^{(1)}, \ldots, y^{(n)}$, the goal of linear regression algorithms is to estimate the regression coefficients $\mathbf{w}$. Many options exist for estimating the coefficients, each with different properties. However, not all are suited for our problem, so choice of algorithm requires care. For example, many traditional statistical regression algorithms (such as ordinary least squares) are only applicable when the number of input vectors is at least the number of input variables (i.e., $p \leq n$). In contrast, in many of our problems, we have many input variables (e.g., the list of websites to try) and significantly fewer input vectors (i.e., $p \gg n$).

Sparse linear regression is a better fit for our problem. It is designed to estimate $\mathbf{w}$ specifically for cases where $p \gg n$, but the regression coefficients $\mathbf{w}$ are assumed to be sparse—i.e., have only a few non-zero entries. This last condition makes the assumption that only a few input values will, in combination, be correlated with the output. One known algorithm for sparse linear regression is Lasso [28]. Under certain conditions on the n input vectors (which we discuss XXX where XXX), Lasso has be proven to estimate $\mathbf{w}$ accurately (with bounded error) as long as $n \geq O(k \log p)$, where k is the number of non-zero entries in $\mathbf{w}$ [5]—i.e., the number of input variables potentially correlated with the output.

The linear regression model is not always a suitable model. In particular, when the output is binary-valued, a generalized linear model called *logistic regression* is often a better fit. For instance, the output could be an indicator variable of whether a particular ad is displayed to the user on a website. This model posits $\Pr[y = 1] = g(\langle \mathbf{x}, \mathbf{w} \rangle)$, where $g(z) = 1/(1 + e^{-z})$. To estimate $\mathbf{w}$, a variant of Lasso has been developed, called L_1-regularized logistic regression [25]. While the theoretical guarantees for this method are not as established as those for Lasso, empirically studies across multiple domains (e.g., XXX, XXX) have demonstrated it to be effective at estimating sparse regression coefficients.

With these basic tools in place, we next describe how we apply them to the targeting problem in Hubble.

[xxx (From Daniel): Is it necessary to use exact code in Fig.4? Seems like some details are unnecessary.]

Modeling the Targeting Problem. Hubble's correlation algorithms leverage both linear and logistic regression models, as well as two different data models (Fig.3) to perform regressions at different granularities of observation. Fig.4(a) shows the code called to run the regressions for each pair of regression type and data model. Hubble is written in Ruby, but uses an implementation of Lasso in R (glmnet library), a programming language specialized in statistics and data analysis.

[1] While the core system developers are systems researchers, our use of statistical methods was supervised by a co-author expert in statistics.

Profiles	Displays		IPs	Input1	Input2	Input3
	#	bool				
profile1	2	1	ip1	1	1	0
profile2	9	1	ip2	1	0	1
profile3	0	0	ip2	0	1	1

(a) **Simple data model matrix**.

Profiles	Context	Displays		IPs	Input1		Input2		Input3	
		#	bool		C	P	C	P	C	P
profile1	input1	2	1	ip1	1	1	0	1	0	0
profile1	input2	0	0	ip1	0	1	1	1	0	0
profile2	input1	7	1	ip2	1	1	0	0	0	1
profile2	input3	2	1	ip2	0	1	0	0	1	1
profile3	input2	0	0	ip2	0	0	1	1	0	1
profile3	input3	0	0	ip2	0	0	0	1	1	1

(b) **Full data model matrix** that includes context.

Fig. 3: **Hubble's data models.** Examples of matrices for Hubble's data models, on three accounts and three inputs, where IP adresses are followed as uncontrolled variables. Displays are used either as a number (#) in linear regressions, or as a boolean value (bool) in logistic regressions. For the full model, each input has two variables: one to measure the influence of the input's context (C) and one for its influence through the whole profile (P).

```
# familly is :logistic or :linear
# model is :full or :simple
def self.regression(expt_id, output_id,
    training_set, familly, model)
  R.data = matrix(expt_id, output_id,
      training_set, model)
  R.familly = familly
  if familly == :linear
    R.displays = data[:log_displays]
  elsif familly == :logistic
    R.displays = data[:bool_displays]
  end

  R.eval <<EOF
    fit <− cv.glmnet(data, displays,
        familly)
    1se <− coef(fit, fit$lambda.1se)
  EOF
  return select_params(expt_id,
    output_id, profiles, R.1se)
end
```

(a)

```
def self.correct(pvalues)
  R.pvalues = pvalues
  R.eval <<EOF
    adjusted <− p.adjust(pvalues,
        method="BY")
  EOF
  return R.adjusted
end
```

(c)

```
def self.pvalue(data, guesses,
    testing_set)
  # mappings is {profile: inputs},
  # observations {profile: #displays}
  mappings = data[:mappings]
  observations = data[:observations]
  profiles_w_ad =
    observations.select { |_, n| n > 0 }.count
  R.p_random = profiles_w_ad / testing_set.count

  R.right = R.wrong = 0
  testing_set.each do |profile|
    if (mappings[profile] & guesses).count > 0
      # at least one input guessed present,
      # we predict ad presence
      observations[profile] > 0 ? R.right += 1 : R.wrong += 1
    end
  end

  R.eval <<EOF
    r <− binom.test(c(right, wrong), p_random)
    pval <− r$p.value
  EOF
  return R.pval
end
```

(b)

Fig. 4: **Correlation code** to (a) run regression, (b) compute p-values and (c) correct for multiple inferences.

For linear regressions the output variable y is $\log(\#\text{displays}/\#\text{trials})$. We normalize with the number of trials during the whole experiment to avoid given too much weight to inputs we would scrap more often. Taking the log is a common practice from statistics when ...

The two different data models allow Hubble to perform the analysis at a profile or a context granularity. First, the simple data model (Fig.3(a)) measures the number of occurrences of an output at a profile granularity. Each input is then modeled as a vector that indicates if the input was present in the profile. The regression will thus measure the correlation between displays of the output, and the presence of an input in a profile, allowing Hubble to detect targeting.

In some cases however, when the current context of an output (e.g., the currently displayed web page) is meaningful to the output (e.g., and ad), this context gives us finer grain information. The full data model (Fig.3(b)) leverages this information to better detect correlation between inputs and outputs. The displays are counted at the level of a tuple (profile, context input) and each input has two explaining variable. One for context effects, which is a vector that indicates if the input was the current context, and one for profile effects that indicates if the input was present in the profile.

Implementation in R. Mention Spark, too, here, please. Mention that R is used at whichever company.

Statistical Significance and p-values. An important contribution of Hubble's correlation detection is that it provides confidence levels for its inferences (e.g., targeting guesses) via p-values. A p-value is a metric of confidence for statistical inferences that measures how likely observed results are under a null hypothesis. Formally, the p-value is the proportion of the null hypothesis distribution that can give results equal or more extreme than what is observed. The smaller the p-value, the less likely observations are to be seen under the null hypothesis, and the more confident we are that the inference is statistically significant.

For example if Hubble's algorithms detect that ad_1 is targeted on a specific website, we should see this ad more often in profiles that visit this website. This means we should be able to predict in which profile this ad will appear better than random guessing. The null hypothesis is that our predictions are the same as those provided by fair coin tosses; we use as our test statistic the number of correct predictions of ad displays across a set of accounts, and the null distribution is the binomial distribution with success probability equal to $1/2$. Hubble splits the dataset into a training set and a testing set (usually 70% and 30% of the total number of accounts, respectively) before performing the analysis. The training set is used to run the algorithm and detect targeting. We then use the testing set to measure confidence—i.e., run the hypothesis test. We predict that each profile containing a targeted input will be shown the ad. If these predictions are significantly more accurate than what could have been achieved by random guessing (as predicted by the null bionomial distribution), then the p-value will be small (e.g., ≤ 0.05, by convention), and thus will lend confidence that our inference detected a real correlation.

We also test the precision of our predictions, which may be a more relevant quantity than accuracy, for instance, when there is substantial imbalance between the number of accounts that are shown an ad and the number of accounts that are not. In this case, a more suitable the null hypothesis is that the accounts on which we predict that an ad was

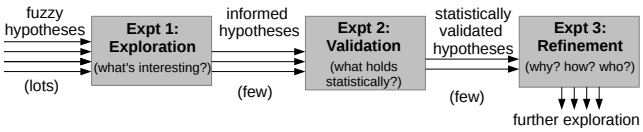

Fig. 5: **Control-Loop Experimental Design.** This is probably not quite te figure we want here – too general and probably not what we can show.

shown are chosen uniformly at random from among the test set. Here, Hubble again splits the dataset as above, and uses the appropriate testing distribution (which in this case is the hypergeometric distribution) to compute the p-value.

Fig.4(b) shows the code that performs this prediction testing to compute the p-values. We leverage the R implementation of a binomial test to compare our ad presence predictions to many random guesses with a probability that corresponds to the fraction of profiles that have the output.

Underlying Assumptions and Known Results.

3.4 Experiment Design

Talk about the various purposes of doing control loop design for a particular tool. Fig.**??** illustrates those varied purposes abstractly. One purpose is to study *why* some effect is happening (e.g., attribution) – e.g., the trackers. Another purpose is to validate (interesting) hypotheses whose confidence is weaker than desired by an auditor (though it is still high enough to believe that there is indeed an effect).

Rules for Experiment Design. Experiment design must fit certain critical rules to be effective with Hubble. First, Hubble assumes that all inputs in a particular experiment can be independently controlled betweeen one another. I.e., the assignment of inputs to profiles must be allowed to be uniform at random. Any relationship that may cause a particular input to will result in violations of causality.

3.5 Accountability Tool Testing

Hubble is flexible enough to support new analysis algorithms, and to be applied to many different questions and hypotheses. It is thus crucial to develop benchmarking techniques to compare new algorithms to existing ones, and to assess the performances of Hubble's analysis building blocks applied to new transparency tools. Because most web services don't provide a ground truth for their targeting, and because manual labelling is tedious and unreliable, we need an automated, reliable benchmarking technique.

Hubble offers transparency developers the ability to test the precision of their tools even in the absence of available ground truth. Once again, Hubble leverages common practices from statistics. To validate or invalidate a prediction, Hubble uses a threshold on the p-values described in §3.3. To be conservative when performing the evaluation we also set aside a bigger proportion of the data for testing compared to regular confidence computation. This gives less training data to make guesses, but ensures more trustworthy p-values.

A common pitfall with p-values arises when performing multiple statistical inferences on the same dataset, as we

E1: Gmail ads targeting experiment:
– registered OnInit().
– **inputs**: list of emails.
– **outputs**: ads on Gmail.
– **uncontrolled_vars**: time, ip.

Fig. 6: **Gmail ads Experiment Design.**

do in Hubble to study multiple ads we collect during our measurements. This causes a *multiple comparison problem*: the risk of false positive increases compared to p-values for a unique inference. Intuitively, by looking at many different hypotheses, we are more likely to find statistical significance by mistake, purely by chance. A common technique to take this effect into account is to do a *false discovery rate* procedure that penalizes p-values to control the expected false positive rate (also called the "false discovery rate"). In Hubble we use the Benjamini-Hochberg-Yekutieli procedure [4] and report penalized p-values to evaluate algorithms. Once again, we leverage an R implementation as shown in Fig.4(c).

To the user, Hubble provides an evaluation mechanism to test the correlation algorithms on new transparency tools, as well as a benchmarking tool on existing datasets to develop new correlation algorithms.

4. Hubble-based Tools

This section show-cases the kinds of web accountability and oversight tools that can be built atop Hubble, and what it takes to build them. To date, we have built two such tools: (1) *TrackTheTracker*, a tool that answers questions about how web trackers use web histories to target ads on the web, and (2) *GObservatory*, a tool that incorporates support for a variety of questions about targeting within and across Google services. To further test our system's flexibility for other kinds of questions, we have additionally gone through the experiment design phases for several targeting questions posed by prior art. This section describes both our implemented tools and designs of prior experiments. Fig.**??** shows the lines of code and experiment designs for our implemented tools.

[xxx Riley and Francis: Please replace Fig. 7 with the figure like the one I sent you. Riley: add your expt design tiles. Please make everything fit into a very compact, preferably single-row figure at the top of the page. Include a LoC table for each tool. Please try to figure out a way to fit everything as compactly as possible. Take a look at how compact the architecture figure is. That's how I want yours to be. Minimize white spaces, they are very costly!]

4.1 TrackTheTracker (TTT)

TTT builds upon Hubble to investigaate the following question: "How do web trackers target ads on a user's brows-

E1 → E2:

E1: Youtube cross service expt:
– registered OnInit().
– **inputs**: list of youtube searches.
– **outputs**: ads on extern websites.
– **uncontrolled_vars**: time, ip.
– **const**: extern websites

E2: Refinement experiment:
– registered OnNewHypothesis(ad, s_in, confidence) if confidence >= 99%.
– let: s_in = youtube search ad targets; w_out = website where ad appears.
– **inputs**: Youtube video ads on w_in.
– **outputs**: ads on w_out.
– **uncontrolled_vars**: time, ip.
– **const**: always visit w_in, w_out.

Fig. 7: **Youtube → outside Google Experiment Design.** E1 is a large-scale survey experiment to find cross-sevice targeting of ads. E2 is a focused refinement experiment to find if video ads on Youtube are used (and if so, which) to target ads.

ing history?" Fig.**??** shows TTT's workflow, which consists of three Hubble experiments: E_1 is a broad survey experiment to find targeted ads on interesting websites; E_2 is a smaller experiment to validate the ads hypothesized as targeted in the survey and prune out untargeted ad and websites; E_3 is a refinement experiment to determine which trackers contributed to the targeting of ads cross-domain that the survey discovered and the validation confirmed. We describe each in turn.

E_1 aims to find promising ads and targeted websites on which to conduct the tracker targeting data collection. In E_1 each input is a website to be surveyed, and there can be hundreds or thousands of these websites. Each website is assigned to a profile with probability 0.5 using the default Hubble profile generation capability; different profiles will have different sets of websites assigned, encoding different browsing histories. The `data collection worker` assigned to a profile drives a headless browser using Selenium [**?**] to visit the pages on each input site assigned to it; it records all display ads observed on each site as Hubble outputs. To detect ads on arbitrary pages, we leverage a version of AdBlock that reports any identified ad but does not disable it. In addition to collecting display ads, the `data collection worker` also records all trackers observed on each site for use in future experiments in the workflow. To detect trackers, we leverage TrackingObserver, a tracking blocker plugin []. TTT uses Hubble's default statistical correlation methods (§**??**) to obtain hypotheses about which output display ads target which input sites.

[xxx Yannis: please update with your validation methodology.] E_2 aims to validate the targeting hypotheses from E_1 in a more controlled environment. E_2 creates groups of profiles where each group consists of the site targeted by an ad and all of the sites on which that ad appears. Each group will have the size of the group plus 20 profiles where each site is assigned to a profile with probability 0.5. The data collection and analysis follow the same procedure as E_1. The ads and their respective groups of site validated as targeted in E_2 will be used in E_3.

E_3 is similar to the first two experiments and uses the same groups of sites as E_2 but instead of using sites as inputs E_3 uses the trackers collected in E_1. For each group of sites TTT takes the union of all of the trackers observed on those sites and creates the magnitude of that set plus twenty accounts for each group. Using the standard Hubble assignment mechanism each tracker is assigned to a profile with probability 0.5. The `data collection worker` used in E_3 drives blocks all trackers not assigned to that profile. It then drives a headless browser to all sites in that group and collects all observed ads as outputs. TTT uses the Hubble's default statistical methods to determine which ads target which trackers.

§**??** shows results of using TTT in measurements of tracker-based ad targeting with 100 input websites selected based on Alexa popularity, 10 pages per website, and hundreds of trackers.

4.2 GObservatory

A number of investigative journalists have asked us a number of questions related to how various massive services, such as Google and Facebook, are using the huge piles of information they have about us to target various classes of populations. We believe that Hubble can be leveraged to investigate such questions more easily and more reliably than ever before. More broadly, we believe that there is great need for comprehensive oversight tools to monitor data flows within and across major services. As a show-case for Hubble, we have begun to build a "data observatory" for Google, *GObservatory*, which integrates experiment designs for investigating a variety of questions about Google targeting: How are Gmail ads targeted on users' inbox contents? How does Youtube target video recommendations based on a user's histories? How does Google Search target the ads? And is Youtube data used to target ads outside of Google services (e.g., for ads on the web)? Many more questions could be potentially supported, and we hope that we and others can integrate more in the future.

To answer these questions, we have built data collectors for each service, which scrape ads from Gmail, Google Search, and YouTube, and recommendations from YouTube. We also leverage TTT's scraping of ads on the web (based on AdBlock) to collect Google ads on the web that might use information from Google accounts to target (based on AdBlock). Scraping takes some engineering effort to encode (especially when no standard tool is applicable, such as AdBlock), but is conceptually simple. For each question, we have defined a Hubble experiment workflow; support for different questions was implemented at different stges of the Hubble design, hence the kinds of experiment designs we implemented for each varies. Fig.6 and Fig.7 show the experiment designs for the Gmail and YouTube questions. We discuss here only Gmail, the simplest (and oldest) of the designs, for which we include results in §**??**.

For Gmail, we implemented a one-level experiment (Fig.6) to determine which emails are used to target ads in Gmail. We wished to extend it to multiple stages, but

half-way during our development, Gmail shut down its ad service. We did get to monitor Gmail ads targeted and their targeting for 30 days before it was suddenly shut down, and present surprising post-mortem results in §**??**. This (now obsolete) feature of GObservatory was inspired by XRay [**?**], however, as §**??** shows, our Hubble-based implementation is much more precise at scale and offers solid statistical guarantees of its predictions.

4.3 Supporting Prior Studies

[xxx Augustin.] Write the followings. Please insert code that shows how you would model each.

Personalization of News & Search. [11] [30] `encore.noise.gatech.edu/faq.html` Please look at the Bobble paper and try to model the questions they ask with our own API/models. Specify the differences in semantics between the results that they would achieve vs. what we would achieve.

Price Discrimination Studies. [12] [24**?**] [**?**]

Please look at some price discrimination study and do the same. Specify the differences/llimitations of our study compared to theirs. One limitation is that we will not be able to model well continuous outputs like the price. Say that the stats do permit incorporating continuous variables, but right now we only support categorical variables (inputs and outputs).

More that could be mentioned: [7] [21] [3]

5. Hubble Evaluation

5.1 Accuracy on Ground Truth

[xxx Mathias.] Evaluate precision/recall of Hubble against ground truth. The results shown here are exclusively from Amazon, YouTube, and the labeled Gmail experiment. Clearly state that Gmail is manual and potentially faulty, and that all three are from simple-case and small-scale experiments. Conclude that Hubble does reasonably well. Remind that XRay does reasonably well on those workloads, too, and say that the differences come at scale and with complex input structures. But say that scale requires a different kind of evaluation methodology than ground truth. So we evaluate that next.

5.2 Tool Testing Evaluation

[xxx Mathias.] This section validates our proposed evaluation methodology by comparing its conclusions with those one would reach with a ground truth. Hopefully, at least for Amazon and Youtube, it should be the case that the conclusions one would reach by evaluating a tool against ground truth would be very similar to those conclusions one would reach by employing our test methodology.

Now that the evaluation tool is validated, we next evaluate the Hubble at scale.

5.3 Accuracy at Scale

[xxx Mathias.] Precision/Recall of Hubble using our testing methodology. Use the larger and the redundant datasets from Gmail, as well as TTT datasets. The results shown here should reflect the scaling properties of Hubble, and how precision/recall varies with scale.

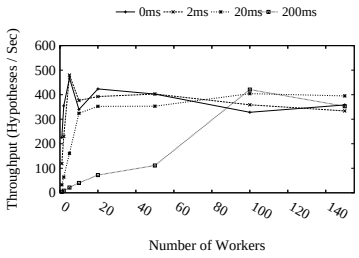

Fig. 8: **Hubble Hypothesis Throughput.** This figure is currently too small. I'll make it bigger later. Plots the number of workers

5.4 Comparison with XRay

[xxx Mathias.] Compare precision/recall of Hubble/stats with XRay's three algorithms. Show the tradeoffs that appear at small scale, and show that Hubble does better at scale and with complex inputs (redundancy). It would be good to include both ground truth results for Amazon and Youtube and results with our testing methodology for Gmail, TTT, and others.

5.5 Performance

[xxx In retrospect the throughput presented in this figure should be normalized to the rate at which we generate notifications. This also needs to be way shorter but I wanted to put something down.] We evaluated Hubble scalability by measuring the number of hypothesis notifications per second that Hubble could produce given a constant stream of inputs and an increasing number of *data analytics workers*. Figure 8 shows the results of this microbenchmark with the number of *data analytics workers* on the x axis and the throughput achieved on the y axis. We implemented this microbenchmark using Hubble's own API. We created a *data collection worker* that calls `addObservation` in a tight loop. We set the observation limit to one so that a notification would be produced on each `addObservation` call and each notification will create a job for a *data analytics worker*. Our *data analytics worker* paused for between 0ms and 200ms to simulate different amounts of analysis time before calling `addHypothesis`. We measured the throughput by the number of hypothesis the system produced per second as we added more hypothesis workers.

The *data collection worker* produced observations at a rate of approximately 500 notifications / second. The experiments with 0 and 2 ms latency achieve their maximum throughput with 5 workers while the experiments with 20 ms and 200 ms latency achieve maximum throughput with 10 and 100 workers respectively. Beyond 100 workers we observed contention on the master for all latencies. We deem this to be acceptable as the observation rate and hypothesis latency will be much lower during an actual deployment. We have observed that most hypothesis generation takes approx-

imately one second and data collection notifications can be tuned by setting the notification limits.

6. Targeting Studies

We used Hubble to run a few example studies of targeting in and across various services. Our goal was not to study one service or question exhaustively, but rather to test our system's ability to answer very diverse questions about various kinds of services. We leave thorough measurement studies of targeting for future work. We used implemented pieces of our GObservatory and TTT tools to pose these questions.

6.1 Targeting of Gmail Ads

As a first example of personal data use, we turn to Gmail which, until November last year, offered personalized advertisments tailored to a user's email content. We selectively placed 300 emails containing single keywords or short phrases to encode a variety of topics, including commercial products (e.g. TV, cars, clothes) and sensitive topics (e.g., religion, sexual orientation, health). Our goal was to study (1) various aspects related to targeted advertisements, such as how frequent they are and how often they appear in the context of the email being targeted (a more obvious form of targeting) versus in the context of another email (a more obscure form of targeting) and (2) whether advertisers are able to target their ads to sensitive situations or special groups defined by race, religion etc. We ran the targeted ad collection for **[xxx 30]** days, from Oct. XXX to Nov. XXX, 2014 when Google shut down the Gmail ad service.

We collected more than 9.1M impressions created collectively by more than 44K unique ads. As expected, the distribution of impressions per ad is skewed: the median ads were observed 9 times in the experiment, while the top 25/5/1% of ads were observed 62/853/2,739 times. We classify an ad as *targeted* if its statistical confidence is high (p-value< 0.05). In our experiment, 5,513 unique ads (12% of all) were classified as targeted, and collectively they are responsible from 43% of all impressions. While we observe that ads classified as targeted are seen more often (352 impressions for the median targeted ads), this could be an artefact of our measurement as most ads seen only occasionally present insufficient evidence and are by default classified as "non-targeted."

[xxx (RG) I don't get this paragraph at all. I removed the statistical method sentence b/c I thought it came out of the blue. My guess is that by now no one is questioning this, why point this out now with such weak evidence? In any case, please try to fix entire paragraph, it's very confusing.] Second, we observe that a non-negligible but small fraction of impressions (7%) for targeted ads appear outside the scope we infer. Interestingly, of the 94% targeted impressions that we correctly predict within scope, a little less than half (49%) appear in the context of an email in scope. This proves that behavioral targeting - where the advertising message is personalized but not related to the current content you are shown - is frequent enough to be the most common form of targeted advertising in this case.

[xxx The abstract mentions a violation of privacy statement. I'd like to keep that as it's an attention drawer (maybe we'll replace the violation word with something weaker. But please add a quote to the privacy statement here.] Finally, we were able to statistically confirm that various personal information, including those related to sensitive topics, can be used by advertiser to deliver their message. A brief summary of ads and associated scope infered can be found in Table **??**. Information about a user's health, race, religious affiliation or religious interest, sexual orientation, or difficult financial situation, all generate targeted advertisment. As Google explicitly proscribed such practice in its term of service for gmail ads, this proves the importance of transparency tools to remain vigilant against ill-intentioned advertisers. One example illustrates how opaque targeting can subtly circumvent current regulation: most ad systems forbids commercial message explicitly encouraging the consumption of recreational drugs, especially if that message is addressed at teenagers, but should an ad promoting equipment for indoor growth of plants be allowed to target email mentioning "cannabis", while its name suggests that it caters to a student audience? While we think those examples are proof of necessary investigation, we would like to point out that those violations are needles in a haystack (about 50 unique ads with debatable scope, less than 0.02% of the ones we observed). Several topics we have included (e.g., fatal diseases and loss, other religious affiliation) generated not a single ad classified as targeted.

6.2 Targeting in Google Search Ads

[xxx Francis.] We posed similar questions for Google Search ads, with very different results.

Main results:

- We have found no evidence of behavioral targeting on sensitive topics.

- In fact, we found evidence of only short-term historybehavioral targeting. Show a graph of how pairs are associated over time. Of course, it is conceivable that there is missed longer-term targeting on topics that we didn't look at, but mostly its very contextual.

- However, we did find another violation of Google's intentions: used guns. See others (perhaps counterfeit goods). We found a number of these examples, which suggests that Google might need a bit of revision on their filtering.

6.3 Targeting of Google Ads on the Web

[xxx Francis.] We also wanted to see whether Google takes data from within our acounts and uses it to target ads outside its services' context. So we looked forany evidence of contamination from Google services to ads, and how it happened. We found that Youtube searches are used to target ads outside. The way it happens is quite interesting – through the Youtube ads themselves.

	email subject & text	ads url & text	Results
Race	**african american** african american african	spokeo.com/Uncover.Scammer.Profiles Who Really Scammed You? Search Their Email Address Fast See Social Profiles & Pics Now!	p<0.05 for 2 days N/A % in context N/A % out context N/A % out scope
	native native american indian native american american indian	www.executivedodgejeep.com Ram 1500 - $48,460 2015 Ram 1500 under $50k with 0 miles!	p<0.001 for 1 day N/A % in context N/A % out context N/A % out scope
Religious affil.	**mormon** mormon mormon	genealogy.com/Family+History Family History Search 1) Simply enter their name. 2) View their family history now!	p<0.02 for 4 days 81 % in context 18 % out context 1 % out scope
	I found God [...] a revelation last night: God talked to me! [...] joining your Church.	www.schwab.com/franchise Open A Schwab Franchise You Built Your Success, Now Own It! Become A Charles Schwab Franchisee.	p<0.01 for 4 days 28% in context 49% out context 22% out scope
Orient.	**Gay** gay homosexual homosexual gay gay lesbian	www.gosoftwear.com MEN Underwear/Workout Underwear, swimwear, Go Natural American Jock, Waist Eliminator	p<0.05 for 4 days N/A % in context N/A % out context N/A % out scope
Health	**Feeling bad** [...] I feel pretty sad, depressed, [...] soon hit rock bottom need help!	www.universities.com Art Study University Search Online And Campus Colleges. Find Top Schools & Degree Programs!	p<0.001 for 1 day N/A % in context N/A % out context N/A % out scope
	cancer advice [...] you dealt well with cancer [...] enduring cancer. Thank you!	www.kidneyregistry.org Kidney Transplant Options My donor is incompatible Paired Exchange gets a better match	p<0.0005 for 1 day 95% in context 0% out context 5% out scope
Recreat. Drugs	**cannabis** cannabis legalisation cannabis legalisation	www.dormgrow.com Grow Green with G8LED High Times Magazine Award Winner High Performance LED Grow Light	p<0.005 for 2 days 100% in context 0% out context 0% out scope
	cannabis cannabis legalisation cannabis legalisation	www.befrugal.com/KFC Printable KFC Coupons Print Free Coupons for KFC. Latest Coupons - Print, Eat & Save!	p<0.0005 for 1 day 100% in context 0% out context 0% out scope

Fig. 9: Personalization of ads within the Gmail service

6.4 Tracker-Based Ad Targeting

[xxx Riley/Yannis.] [xxx IT'S VERY URGENT THAT WE GET SOME HYPOTHESES HERE AND TEST THEM.]

7. Related Work

Although the topic of web transparency and accountability has garnered significant attention in several communities (e.g., security, networking, and machine learning), hardly any systems work exists, that focuses on building generic, scalable, and principled systems to make accountability and oversight more amenable on the giant, ever-changing web. Hubble is the first infrastructure system that meets a comprehensive list of practical requirements, including scale, extensibility, sound metrics for confidence levels in the results, and real-time investigations and validations of interesting hypotheses.

Targeting Measurement Studies. [xxx Augustin.] Price discrimination studies, news/search bubble studies, ad targeting studies, etc.

Web Transparency and Accountability. [xxx Augustin.] XRay, OpenWPM, Anupam's stuff, Bobble. HTTPA (Tim Berners Lee).

Further afield, much prior work has focused on *data collection transparency*, which seeks to reveal what information services *collect* about the users [1, 15–18, 23, 26, 27]. The work is complementary to ours, which seeks to reveal what web services *do* with the data they collect.

Algorithmic Transparency. [xxx Daniel.] People in ML community seem to have started to talk about algo transparency (algos that are built with transparency in mind). Talk about conceptual approaches in which the "world" can be made inherently more transparent, which might in the end help the building of tools like Hubble. HTTPA is also related to this. Same observation as for Learning theory (don't go into deep theory – OS folks will get lost).

Learning Theory. [xxx Augustin.] Whetever Rocco is doing. Present the basic goals of this field, major known results. Please don't go into details of the theory, remember you're talking to OS folks.

Related Statistical Methods. Our methods for statistical experimental design and analysis draw from the subjects of *compressed sensing* [6, 8], *sparse linear regression* [5, 28], and *sparse signal recovery* [9, 10]. The experimental setups we consider correspond to sensing matrices that satisfy certain analytic properties that permit robust recovery of sparse signals. In Hubble, these signals correspond to the hypothesized targeting effects we subsequently test and validate, and they are sparse when the targeting effects only depend on a small number of variables (e.g., **[xxx e-mail types]**). Our work has so far only considered simple and non-adaptive methods for generating these sparse hypotheses; there is substantial room for further exploration. First, we may support different analysis methods that improve over standard methods (like Lasso) in either computational efficiency [20, 29] or in statistical power [32]. These alternative methods may provide different computational/statistical trade-offs that can be assessed by the application developer. Many of these alternative methods also naturally fit in frameworks for distributed computation such as Spark [31] and GraphLab [22]. Second, we may use *adaptive methods* [13, 14] to potentially reduce the data collection requirements. Such methods use the outcomes of initial experiments to decide which experiments are best to conduct next. Finally, we may also explore (adaptive) recovery of *non-linear hypotheses* that are commonly used in machine learning [2]. Indeed, the theoretical framework of *learning with membership queries* may provide useful insights into experimental designs for identifying very rich classes of targeting mechanisms.

Anything Else. [xxx Augustin?]

8. Conclusions

Preserving privacy XXX a challenging arms-race against powerful economic drivers and human factors. What remains is

References

[1] ACAR, G., EUBANK, C., ENGLEHARDT, S., JUAREZ, M., NARAYANAN, A., AND DIAZ, C. The Web never forgets: Persistent tracking mechanisms in the wild. *CCS '14: Proceedings of the 21st ACM conference on Computer and com-*

munications security (2014).

[2] ANGLUIN, D. Queries revisited. In *Algorithmic Learning Theory*, N. Abe, R. Khardon, and T. Zeugmann, Eds., vol. 2225 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 12–31.

[3] BARFORD, P., CANADI, I., KRUSHEVSKAJA, D., MA, Q., AND MUTHUKRISHNAN, S. Adscape: Harvesting and Analyzing Online Display Ads. *WWW '14: Proceedings of the 23nd international conference on World Wide Web* (Apr. 2014).

[4] BENJAMINI, Y., AND YEKUTIELI, D. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics* (2001), 1165–1188.

[5] BICKEL, P. J., RITOV, Y., AND TSYBAKOV, A. B. Simultaneous analysis of lasso and dantzig selector. *Ann. Statist. 37*, 4 (08 2009), 1705–1732.

[6] CANDÈS, E. J., ROMBERG, J. K., AND TAO, T. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory 52*, 2 (2006), 489–509.

[7] DATTA, A., TSCHANTZ, M. C., AND DATTA, A. Automated Experiments on Ad Privacy Settings: A Tale of Opacity, Choice, and Discrimination. *arXiv.org* (Aug. 2014).

[8] DONOHO, D. L. Compressed sensing. *IEEE Transactions on Information Theory 52*, 4 (2006), 1289–1306.

[9] GILBERT, A., AND INDYK, P. Sparse recovery using sparse matrices. *Proceedings of the IEEE 98*, 6 (June 2010), 937–947.

[10] GILBERT, A. C., STRAUSS, M. J., TROPP, J. A., AND VERSHYNIN, R. One sketch for all: Fast algorithms for compressed sensing. In *39th ACM Symposium on Theory of Computing* (2007), pp. 237–246.

[11] HANNAK, A., SAPIEZYNSKI, P., KAKHKI, A. M., KRISHNAMURTHY, B., LAZER, D., MISLOVE, A., AND WILSON, C. Measuring personalization of web search. In *WWW '13: Proceedings of the 22nd international conference on World Wide Web* (May 2013), International World Wide Web Conferences Steering Committee.

[12] HANNAK, A., SOELLER, G., LAZER, D., MISLOVE, A., AND WILSON, C. Measuring Price Discrimination and Steering on E-commerce Web Sites. *IMC '14: Proceedings of the 14th ACM SIGCOMM conference on Internet measurement* (2014).

[13] HAUPT, J. D., BARANIUK, R. G., CASTRO, R. M., AND NOWAK, R. D. Compressive distilled sensing: Sparse recovery using adaptivity in compressive measurements. In *Proceedings of the 43rd Asilomar Conference on Signals, Systems and Computers* (2009), pp. 1551–1555.

[14] INDYK, P., PRICE, E., AND WOODRUFF, D. P. On the power of adaptivity in sparse recovery. In *IEEE 54th Annual Symposium on Foundations of Computer Science* (2011), pp. 285–294.

[15] KRISHNAMURTHY, B. I know what you will do next summer. *ACM SIGCOMM Computer Communication Review* (2010).

[16] KRISHNAMURTHY, B., MALANDRINO, D., AND WILLS, C. E. Measuring privacy loss and the impact of privacy protection in web browsing. In *Proc. SOUPS* (New York, New York, USA, 2007), ACM Press, pp. 52–63.

[17] KRISHNAMURTHY, B., AND WILLS, C. Privacy Diffusion on the Web: A Longitudinal Perspective. In *Proc. ACM WWW* (New York, New York, USA, 2009), ACM Press, p. 541.

[18] KRISHNAMURTHY, B., AND WILLS, C. E. On the leakage of personally identifiable information via online social networks. *SIGCOMM Computer Communication Review 40*, 1 (Jan. 2010).

[19] LECUYER, M., DUCOFFE, G., LAN, F., PAPANCEA, A., PETSIOS, T., SPAHN, R., CHAINTREAU, A., AND GEAMBASU, R. XRay: Enhancing the Web's Transparency with Differential Correlation . In *23rd USENIX Security Symposium (USENIX Security 14)* (San Diego, CA, 2014), USENIX Association.

[20] LIN, D., FOSTER, D. P., AND UNGAR, L. H. VIF regression: A fast regression algorithm for large data. *Journal of the American Statistical Association 106*, 493 (2011), 232–247.

[21] LIU, B., SHETH, A., WEINSBERG, U., CHANDRASHEKAR, J., AND GOVINDAN, R. AdReveal: improving transparency into online targeted advertising. In *HotNets-XII: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks* (Nov. 2013), ACM Request Permissions.

[22] LOW, Y., BICKSON, D., GONZALEZ, J., GUESTRIN, C., KYROLA, A., AND HELLERSTEIN, J. M. Distributed GraphLab: A framework for machine learning and data mining in the cloud. *Proc. VLDB Endow. 5*, 8 (Apr. 2012), 716–727.

[23] MAYER, J. R., AND MITCHELL, J. C. Third-Party Web Tracking: Policy and Technology. *Security and Privacy (S&P), 2012 IEEE Symposium on* (2012), 413–427.

[24] MIKIANS, J., GYARMATI, L., ERRAMILLI, V., AND LAOUTARIS, N. Detecting price and search discrimination on the internet. In *HotNets-XI: Proceedings of the 11th ACM Workshop on Hot Topics in Networks* (Oct. 2012), ACM Request Permissions.

[25] NG, A. Y. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning* (2004).

[26] REISMAN, D., ENGLEHARDT, S., EUBANK, C., ZIMMERMAN, P., AND NARAYANAN, A. Cookies that give you away: Evaluating the surveillance implications of web tracking. *Princeton University* (Apr. 2014).

[27] ROESNER, F., KOHNO, T., AND WETHERALL, D. Detecting and defending against third-party tracking on the web. In *NSDI'12: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (Apr. 2012), USENIX Association.

[28] TIBSHIRANI, R. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B 58* (1994), 267–288.

[29] XIAO, L. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research 11* (2010), 2543–2596.

[30] XING, X., MENG, W., DOOZAN, D., FEAMSTER, N., LEE, W., AND SNOEREN, A. C. Exposing Inconsistent Web Search Results with Bobble. *Passive and Active Measurements Conference* (2014).

[31] ZAHARIA, M., CHOWDHURY, M., FRANKLIN, M. J., SHENKER, S., AND STOICA, I. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing* (Berkeley, CA, USA, 2010), HotCloud'10, USENIX Association, pp. 10–10.

[32] ZHANG, T. Adaptive forward-backward greedy algorithm for learning sparse representations. *IEEE Transactions on Information Theory 57*, 7 (July 2011), 4689–4708.