iCheck: Computationally Combating "Lies, D—ned Lies, and Statistics"

You Wu[†] Brett Walenz[†] Peggy Li[†] Andrew Shim[†] Emre Sonmez[†]
Pankaj K. Agarwal[†] Chengkai Li[‡] Jun Yang[†] Cong Yu[§]
[†]Duke University, [‡]University of Texas at Arlington, [§]Google Research

ABSTRACT

Are you fed up with "lies, d—ned lies, and statistics" made up from data in our media? For claims based on structured data, we present a system to automatically assess the quality of claims (beyond their correctness) and counter misleading claims that cherry-pick data to advance their conclusions. The key insight is to model such claims as parameterized queries and consider how parameter perturbations affect their results. We demonstrate our system on claims drawn from U.S. congressional voting records, sports statistics, and publication records of database researchers.

1 Introduction

Claims of "fact" are made from data constantly—by journalists, politicians, lobbyists, public relations specialists, sports fans, etc. Wherever numbers and data are involved, they can be laden with so-called "lies, d—ed lies, and statistics." Consider the following.

Example 1 (drawn from [5, 6])**.** ***Giuliani's adoption claim*** *(from* factcheck.org*). During a Republican presidential candidates' debate in 2007, Giuliani claimed that "adoptions went up 65 to 70 percent" in the New York City "when he was the mayor." More precisely, the comparison is between the total number of adoptions during 1996-2001 and that during 1990-1995 (he was in office 1994-2001). The claim "checks out" according to data, but why does it compare these two particular periods? As it turns out, the underlying data actually show that adoption began to slow down in 1998, a trend that continued through 2006. Lumping data into the two six-year periods masks this trend. Comparing the beginning and the end of his tenure would have yielded only a 17% increase.*

One-of-the-few NBA players claim*. "Only 10 players in NBA history had more points, more rebounds, and more assists per game than Sam Lacey in their career." There are nearly 4000 players in NBA history, so "one of the 11" sounds impressive. However, we can claim the exact same or stronger for 112 other players (that no more than 10 players dominate them in these three stats).*

Vote correlation claim *(from* factcheck.org*). A TV ad in the 2010 elections claimed that Jim Marshall, a Democratic incumbent from Georgia, "is a long way from Nancy Pelosi," as he "voted the same as Republican leaders 65 percent of the time." This comparison was made with Republican Leader John Boehner over the votes in 2010. If we start the comparison from 2007, however, the number would have been only 56 percent, which is not very high considering that even the Democratic Whip, Jim Clyburn, voted 44 percent of the time with Boehner during that period.*

SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.

ACM 978-1-4503-2376-5/14/06 ...$15.00.
http://dx.doi.org/10.1145/2588555.2594522.

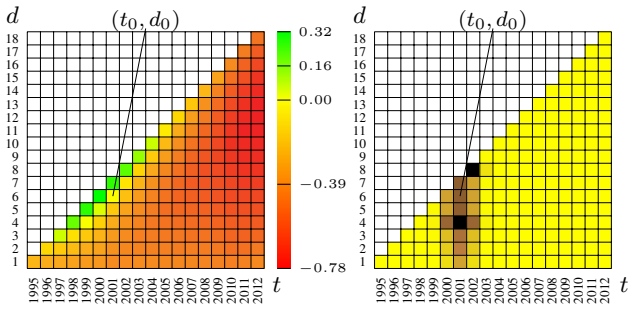
Today's highly connected world has made it ever easier to perpetuate lies; at the same time, the movement of "democratizing data" has also made it possible to counter these lies with an increasing array of publicly available data. Thus, there is pressing demand for data-driven fact-checking today. But checking claims—even those based on clean, structured data—is challenging. For example, all claims above are actually "correct," but they present rather misleading views of data. Thus, we must go beyond correctness to find better measures of claim quality to catch lies. Even before we can check a claim, however, we may have to clarify it: many claims, such as Giuliani's and Marshall's above, are stated vaguely (often intentionally). Finally, to help explain to the public why a claim has a low quality, we need "counterarguments": e.g., for Giuliani's claim, we highlighted that "*Comparing the beginning and the end of his tenure would have yielded only a 17% increase*."

Organizations such as *factcheck.org* and *PolitiFact.com* rely on their expert editorial staff to check claims. However, manual approaches are difficult to scale because of the demand on human expertise and effort. *FactMinder* [4] is a tool that assists fact-checkers in annotating text, extracting entities, linking sources, and collaboratively building a knowledge base. *Truth Goggles* (*truthgoggl.es/demo.html*) and *Dispute Finder* [3] detect claims on the Web that have already been checked or refuted by authoritative sources. However, computational tools for checking claims directly using data are still sorely lacking.

To fill this void, we present a system called *iCheck*, which demonstrates that we can, in fact, cast the problem of checking claims based structured data as a computational one. With a general modeling framework and an extensible system, iCheck rates claims of various kinds using quality measures that capture the intuition behind the type of analysis seen in Example 1. iCheck can also automatically "reverse-engineer" a vague claim to recover missing details, and come up with counterarguments to low-quality claims. Interestingly, the same machinery that allows iCheck to combat lies can also be used to generate "interesting" claims including lies. In the demonstration, we will pit iCheck against itself over U.S. congressional voting records, sports statistics, and publication records of database researchers, letting users generate claims in these domains, and then offering a healthy dose of reality check.

2 Modeling Overview

The key intuition behind our approach is remarkably simple. Think of a claim as a parameterized query—we can learn a lot about it by



(a) Relative result strength (b) Parameter sensibility

Figure 1: (From [6].) Perturbing t and d in Giuliani's claim while fixing $w = 6$. The data became available in 1989, so $t - d - w \geq 1988$.

"perturbing" its parameters.[1] To illustrate, Giuliani's claim in Example 1 is parameterized by (w, t, d), where $w = 6$ is the length of the aggregation window (in years), $t = 2001$ is the end of the second window, and $d = 6$ is the distance between the two windows being compared. Perturbing the parameter setting to $(1, 2001, 8)$, for example, changes the increase to merely 17%. As another example, the one-of-the-few claim in Example 1 is parameterized by the player (Sam Lacey for the original claim). Perturbing this parameter to other players reveals how unique Sam Lacey is. Finally, the vote correlation claim in Example 1 is parameterized by both the period of comparison and the legislators being compared.

The QRS Framework We proposed this framework in [6] to formalize various claim quality measures and formulate problems in finding and checking claims. Briefly, we model a *claim template* with m parameters as a function $q : \mathcal{P} \to \mathcal{R}$ (given a fixed database instance). Here, $\mathcal{P} \subseteq P_1 \times \cdots \times P_m$ is the *parameter space*, where P_i is the domain of the i-th parameter. Given a parameter setting $\vec{p} = (p_1, \ldots, p_m)$, $q(\vec{p})$ denotes the result of the query instantiated with $\vec{p}$. $\mathcal{R}$ is the domain of results. The points $\{(\vec{p}, q(\vec{p})) \mid \vec{p} \in \mathcal{P}\}$ in $\mathcal{P} \times \mathcal{R}$ form q's *query response surface* (*QRS*).

Intuitively, a fact-checker explores the QRS by trying various perturbations to see how they weaken or strengthen the original claim. Not all perturbations are equally useful to investigation—useful perturbations should be "relevant" (i.e., they stay within the context of the claim) and "natural" (e.g., changing "58 months" to "5 years" makes the claim more concise), and should significantly "weaken" the original claim. With this intuition, we introduce two scoring functions as helpers (see [6] for additional details):

- A *(relative) result strength function* $\mathsf{SR} : \mathcal{R} \times \mathcal{R} \to \mathfrak{R}$. $\mathsf{SR}(r; r_0)$ captures how much $r \in \mathcal{R}$ (result for a perturbed claim) deviates from a *reference* result $r_0 \in \mathcal{R}$ (usually the result of the original claim). Negative $\mathsf{SR}(r; r_0)$ means the claim is weakened.
- A *(relative) parameter sensibility function* $\mathsf{SP} : \mathcal{P} \times \mathcal{P} \to \mathfrak{R}$. $\mathsf{SP}(\vec{p}; \vec{p}_0)$ captures how "sensible" a parameter setting $\vec{p} \in \mathcal{P}$ is with respect to a *reference* setting $\vec{p}_0 \in \mathcal{P}$. High $\mathsf{SP}(\vec{p}; \vec{p}_0)$ means $\vec{p}$ is natural and relevant to the original claim context.

As an example, consider again Giuliani's claim. Since the full space of (w, t, d) is difficult to visualize, Figure 1a fixes $w = 6$ and plots the result strength function over (t, d) relative to the original claim at (t_0, d_0). While some perturbations (near the diagonal, shown in greener colors) strengthen the original claim, most perturbations (shown in redder colors) weaken it. However, not all perturbations are equally sensible. Figure 1b plots the relative parameter sensibility function; a darker color indicates higher sensibility. Sensibility scores generally drop around the original parameter setting (they become less relevant), but bumps occur at $d = 4$ and 8 (they are more natural because a mayor's term is four years). High-sensibility regions of $\mathcal{P}$ are more useful in checking the claim.

Measures of Claim Quality With the QRS framework, we can now define claim quality measures formally. As examples, we present the two measures used in our demonstration scenarios. We assume a finite and discrete $\mathcal{P}$ (generalization to the infinite and continuous case is straightforward).

- *Robustness* is defined as
$$\exp(-\textstyle\sum_{\vec{p} \in \mathcal{P}} \mathsf{SP}(\vec{p}; \vec{p}_0) \cdot (\min(0, \mathsf{SR}(q(\vec{p}); r_0)))^2).$$
Intuitively, it is based on the mean squared deviation of randomly perturbed claims (we consider the deviation to be 0 if the perturbed claim is stronger than the original). Robustness of 1 means all perturbations result in claims that are at least equally strong; robustness close to 0 means the original claim can be easily weakened. Giuliani's claim has low robustness.
- *Uniqueness* is defined as $\sum_{\vec{p} \in \mathcal{P}} \frac{1}{|\mathcal{P}|} \cdot \mathbf{1}(\mathsf{SR}(q(\vec{p}); r_0) < 0)$, where $\mathbf{1}(\cdot)$ is the indicator function. Intuitively, uniqueness is the fraction of parameter settings that yield results weaker than the original claim. Low uniqueness means it is easy to find perturbed claims at least as strong as the original. The one-of-the-few claim in Example 1 about Sam Lacey has low uniqueness.

Finding Counterarguments Besides quantifying claim quality, we can, with the help of QRS, cast the task of *finding best counterarguments* also as a computational problem. To counter a low-uniqueness claim, we show a (large enough) sample of the parameter settings that yield claims at least as strong as the original. To counter a low-robustness claim, we find the best counterarguments by bi-criteria optimization over the QRS: given a bound $-\delta$ on the deviation from the original claim result r_0, find the perturbed parameter setting $\vec{p}$ from the original setting $\vec{p}_0$ such that $\mathsf{SR}(q(\vec{p}); r_0) \leq -\delta$ and $\mathsf{SP}(\vec{p}; \vec{p}_0)$ is maximized. The dual and Pareto-optimal formulations are also plausible.

Reverse-Engineering Vague Claims With QRS, we can also cast the problem of reverse-engineering vague claims as finding the "isoline" of the QRS that contains the original claim result. With the help of parameter sensibility function SP, we can find the recovered claim that is the most sensible. More precisely, suppose the original claim has result r_0, and let $\vec{p}_0$ be a reasonable parameter setting capturing the claim context (without worrying about whether $q(\vec{p}_0) = r_0$). For example, Giuliani's adoption claim specifies none of the three parameter values; we simply pick $\vec{p}_0 = (w_0, t_0, d_0) = (1, 2001, 8)$, reflecting that Giuliani was in office 1994–2001. Then, the most sensible recovered parameter setting $\vec{p}$ is the one that maximizes $\mathsf{SP}(\vec{p}; \vec{p}_0)$ while satisfying $\mathsf{SR}(q(\vec{p}); r_0) \approx 0$ (with an error bound allowing for the case when the original claim states its result vaguely).

Making Claims The QRS framework also allows us to formulate the task of making claims as computational problems. Given a claim template and a dataset, we can find all claims with high quality measures (such as uniqueness); we tackled this problem for one-of-the-few claims in [5]. Alternatively, further given an object of interest, we can find all parameter settings involving this object with high parameter sensibility and high result strength. This latter problem is the same as finding counterarguments to low-robustness claims discussed earlier, except it reverses the sign of the objective function for result strength. Note that the optimization criteria in this case do not include any quality measures, so we can generate "lies," which are useful to our demo (see Section 4 for details).

[1] In fact, we can also consider perturbing the data over which the query is evaluated, or even consider parameter and data perturbations jointly. Our current system, however, focuses only on perturbing parameters. Therefore, we will present our framework only in terms of parameter perturbations.

3 System Overview

Having seen how to model fact-checking tasks as computational problems, we now discuss how to implement this computational approach. A main challenge is the diversity of claims and domains. Such diversity manifests itself in both computation and modeling—different claim templates may require different algorithms to process efficiently, and knowledge about different domains needs to be captured by domain-specific scoring functions. Implementing each new claim template or applying it to each new domain completely from scratch will be too labor-intensive. Thus, our overarching design goal is to make the iCheck architecture general and extensible.

iCheck offers "pay-as-you-go" extensibility. The idea is that new claim templates and domains can be added with little effort to enable basic support, but if needed, better modeling and algorithms can be achieved with extra effort. We implement this idea by factoring out common domain properties, model components, and algorithmic ideas, and providing reasonable "defaults" that can be overridden by more sophisticated "plugins." In the following, we highlight how various aspects of iCheck implement this idea.

Algorithmic Framework In [6], we developed an algorithmic framework consisting of a suite of "meta" algorithms with pluggable low-level building blocks. The baseline meta algorithms are the most general and assume no special building blocks. More efficient meta algorithms assume building blocks that enable enumeration of points in the parameter space $\mathcal{P}$ in the order of sensibility. The most efficient meta algorithms use building blocks that enable a divide-and-conquer approach over $\mathcal{P}$. Thus, new templates are supported by the baseline algorithms with little effort upfront; higher efficiency can be achieved with more sophisticated building blocks, without re-implementing the high-level algorithms.

As an example, for Giuliani's claim in Example 1, consider the problem of finding the strongest counterarguments given a sensibility threshold. The *baseline* algorithm simply examines the 3-d parameter space $\mathcal{P}$ and tries every (w, t, d) setting. This algorithm is acceptable for this particular claim, because the time series of adoption numbers is small and the number of parameter settings is limited. In general, however, faster algorithms may be needed.

The algorithm based on *ordered enumeration* enumerates parameter settings in decreasing order of sensibility, and stops as soon as it reaches the sensibility threshold. For Giuliani's claim, the parameter sensibility function is a monotone combination of three per-dimension scores (for w, t, and d). Therefore, we only need to provide a building block for each dimension that enumerates the parameter values in this dimension in decreasing order of per-dimension score. The meta algorithm combines these building blocks to enable ordered enumeration in $\mathcal{P}$.

Finally, the *divide-and-conquer* meta algorithm works by dividing the parameter space into a number of *zones*, and solving the optimization problem in each zone. To enable this approach for Giuliani's claim, we provide a building block that divides the subset of $\mathcal{P}$ above the sensibility threshold into a series of zones, where each zone is a 2-d eclipse of possible (t, d) values (further bounded by linear constraints) with a given w value. Within each such zone, we can find the strongest counterargument (the parameter setting minimizing the relative strength with respect to the original claim) efficiently by reducing this problem to the well-studied range-maximum query over a static sequence; see [6] for details.

In sum, faster algorithms can be enabled by more sophisticated building blocks that are increasingly specialized for the claim template and domain. For the example above, if we assume that all parameter settings meeting the sensibility threshold fall within a sphere of radius $\tilde{r}$, then ordered enumeration takes $O(\tilde{r}^3 \log \tilde{r})$ time; divide-and-conquer only takes $O(\tilde{r}^2)$ time. In comparison, baseline always takes $O(|\mathcal{P}|)$ time, regardless of how small $\tilde{r}$ is.

Domain Knowledge Toolbox Various functions in our modeling and algorithmic frameworks capture domain knowledge needed to assess claim quality and enable efficient computation. As an example, for Giuliani's claim, the parameter sensibility function SP is the product of three scores, one for each of (w, t, d). The score for d, the distance between the two windows begin compared, further consists of two components: 1) *relevance*, which measures how relevant a perturbed d is to the context of the original claim; and 2) *naturalness*, which captures how intrinsically natural the perturbed d value is. We model relevance of d by $\exp(-(\frac{d-d_0}{\sigma_d})^2)$, where d_0 is the setting used in the original claim, and σ_d is used to control the weight of d's relevance relative to its naturalness and to the scores of other parameters. Intuitively, this relevance function penalizes big perturbations in d to favor counterarguments that are "close" to the original claim in their parameter settings. As for d's naturalness, because the New York City mayor has four-year terms, it is more natural to compare with windows that are multiples of 4 years apart; therefore, we assign higher naturalness to d values that are divisible by 4. For details, see [6].

Setting up such functions is not trivial, and we cannot afford to repeat this process for every parameter of every claim template for every dataset. Fortunately, for most of claims that we have examined, a handful of functional forms work well. For example, most parameters can be scored by a relative relevance term and an absolute naturalness term, like the parameter d discussed above. As another example, most numeric parameters have periodic naturalness scores, though their periods vary (e.g., quantities that are multiples of thousands, hundreds, or tens appear more natural in claims than those are not; numbers of months in the multiples of 12 are more natural). Therefore, iCheck implements a toolbox of such function forms, which user can choose and instantiate for the problem at hand. For the supported forms of sensibility functions, the toolbox also provides the enumeration functions required to enable meta algorithms based on ordered enumeration.

To simplify the task of encoding domain knowledge, iCheck provides a learning-based tool, which presents the user with a series of questions, such as asking the user to judge an alternative claim's strength and sensibility relative to the original claim. The tool then uses the answers to select appropriate function forms for SP and SR, and to train their parameters automatically.

Claim Template Library The same claim template are often applicable to many different scenarios. To avoid re-implementing the same template for different datasets and domains, iCheck maintains an extensible library of *generic* claim templates, which allow different datasets and SP/SR functions to be plugged in.

A generic claim template assumes that essential data are available through a *canonical schema*, which includes relational tables, and, in some cases, user-defined functions. The canonical schema hides the complexities of the actual datasets to which this template may be applied. For example, the vote correlation claim in Example 1 is an instance of the generic *TSS* (*time series similarity*) claim template. The TSS template assumes a table $\texttt{series}(\underline{\texttt{id}}, \underline{\texttt{t}}, \texttt{v})$, where `id` identifies an object of interest (e.g., a legislator), `t` is a timestamp, and `v` is the value associated the object at `t`. The template also assumes a user-defined function $\texttt{match}(\texttt{v}_1, \texttt{v}_2)$ that tests whether two values match. A TSS claim states that two objects' values match for some fraction of the time over a period.

Applying a generic claim template to a particular dataset requires us to define the mapping of actual data to tables defined by the canonical schema, and to supply the required user-defined func-

tions. For example, given a database of the U.S. congressional voting records, to check claims about Representatives, we would define `series` as a view returning all relevant votes in the House; we would also define $\texttt{match}(\texttt{v}_1, \texttt{v}_2)$ to compare recorded votes appropriately (with attention to special values such as "present" or "not voting"). In addition, we need to provide functions to convert a TSS query with parameter values and result to a more human-readable form tailored toward the domain. For example, instead of making generic TSS counterarguments showing raw `t`'s and `id`'s, iCheck will invoke the conversion functions to show sentences talking about vote correlations over periods of dates, with full names of Representatives and URLs for additional information about them.

Generic claim template code specifies default choices of functions used by our modeling and algorithmic frameworks. It can also implement optimizations and algorithmic building blocks specific to this claim template. When applying the claim template to a specific scenario, we may find the defaults supplied by the template code already adequate; if so, this new application takes minimal effort. Otherwise, we need to override the defaults, with help from the domain knowledge toolbox described earlier, including the learning-based tool for setting up SP and SR. In any case, based on the functions available, the template code will invoke the most efficient meta algorithm that is enabled.

4 Demo Scenarios and User Interface

We will demonstrate most functionalities of iCheck through a website. This website has been customized for several domains (described further at the end of this section), but the underlying machinery for finding and checking facts remains the same.

The website provides an *object-centric view* that consolidates information about an object of interest, e.g., a U.S. legislator. On this view, a user can browse the list of claims known to iCheck about this object. These claims can be sorted by template, quality (using the measures defined in Section 2), and popularity (based on the level of user activities), allowing the user to spot interesting claims easily. From the object-centric view, the user can manually enter a claim about the object (e.g., from an attack ad against an incumbent) by selecting the claim template and filling out a form. Alternatively, the user can ask the system to come up some claims about this object that are correct, though possibly misleading. For example, the user can ask for claims implying that a legislator is too liberal (by citing vote correlation with well-known liberals).

The *claim-centric view* is where iCheck scrutinizes a claim. This view shows applicable quality measures, the list of counterarguments, as well as a list of "related" claims (whose definition depends on the domain). For vague claims, e.g., the vote correlation claim in Example 1, this view shows the list of possible reversed-engineered claims, each with its own view. Users can comment on the claim-centric views and share claims via social media. We will set up a Twitter hash tag so that conference followers not present at our demo will also be able to get a taste of it.

Besides comments, users can provide feedback in a number of ways. They can vote for the best counterarguments and reversed-engineered claims. We randomly challenge user to gauge the quality of a claim before showing our verdict and evidence; they will then be asked to judge the effectiveness of our fact-checking. User feedbacks will be summarized and shown on relevant views together with measures computed automatically by iCheck.

For those interested in the inner workings of iCheck, we will demonstrate, on a command-line interface, the process of making a new claim template, and applying an existing template to a new dataset (including using the tool that learns appropriate SP and SR through user feedbacks discussed in Section 3).

Domains We have prepared three domains: U.S. congressional voting records, sports statistics, and publication records of database researchers. The congressional voting records from *govtrack.us* contain more than 22 millions votes cast since May 1789 and are updated daily. We focus on vote correlation claims such as the one in Example 1, but additionally consider perturbing the subset of bills on which comparison is made. One useful subset is that of "controversial" bills, on which votes are largely divided between party lines. We also define subsets by legislative topics (e.g., gun control and health care), allowing users to explore "related" claims that zoom in on issues they care about.

For sports, we use NBA and MLB player statistics. From a popular EPSN blog (*espn.go.com/espn/elias*) by Elias Sports Bureau, we have chosen a few frequently used claim templates about players, including the "one-of-the-few" template in Example 1.

Interested in neither politics nor sports? We have a domain just for the SIGMOD audience. Using the DBLP (*dblp.uni-trier.de*) dataset, we allow a user to find impressive-sounding snippets to include in c.v. or tenure and promotion letters, and, if so desired, have a reality check on these snippets. For example, iCheck finds that one of the authors of this paper had a great year of publications—only 8 other researchers have ever managed to publish as much as (or more than) him in one year in all three venues: SIGMOD, VLDB, and ICDE. Of course, iCheck then readily points out that the same claim can be made for 89 other researchers!

5 Conclusion

This work falls under the general theme of *computational journalism* [2, 1], a nascent discipline that harnesses computing to help journalism cope with the challenges and opportunities of the digital age. iCheck is only a first step toward a computational approach to fact-checking. We have not addressed a number of related issues, such as continuously monitoring media for checkable claims, and automatically mapping natural-language claims to known templates. Furthermore, not all lies can be detected automatically by inspecting data and numbers alone; fact-checking in general needs to leverage collective human intelligence. Nonetheless, our demonstration illustrates the potential of computational techniques—in reducing cost and increasing effectiveness—for certain types of investigation tasks with growing importance today, as more structured datasets become available either directly or by information extraction. There has been some work already on computer-aided fact-checking (e.g., [3, 4]) complementing ours; we hope more will join us in tackling this problem of significant societal importance.

Acknowledgments The authors are supported by NSF grants IIS-09-16027, CCF-09-40671, IIS-10-18865, CCF-10-12254, CCF-11-17369, CCF-11-61359, and IIS-13-20357. Addition support comes from ERDC Contract W9132V-11-C-0003, ARO Contract W911NF-13-P-0018, Grant 2012/229 from the U.S.-Israel Binational Science Foundation, HP Labs Innovation Research Awards, and a Google Faculty Research Award. Any opinions, findings, and conclusions in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.

References

[1] Sarah Cohen, James T. Hamilton, and Fred Turner. Computational journalism. *Communications of the ACM*, 54(10):66–71, 2011.

[2] Sarah Cohen, Chengkai Li, Jun Yang, and Cong Yu. Computational journalism: A call to arms to database researchers. In *CIDR* 2011.

[3] Rob Ennals, Beth Trushkowsky, and John Mark Agosta. Highlighting disputed claims on the Web. In *WWW* 2010.

[4] François Goasdoué, Konstantinos Karanasos, Yannis Katsis, Julien Leblay, Ioana Manolescu, and Stamatis Zampetakis. Fact checking and analyzing the Web. In *SIGMOD* 2013.

[5] You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. On "one of the few" objects. In *KDD* 2012.

[6] You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. Toward computational fact-checking. *PVLDB*, 7(7), 2014.