Finding, Monitoring, and Checking Claims Computationally Based on Structured Data

The iCheck/uClaim Team
Brett Walenz, You (Will) Wu, Seokhyun (Alex) Song, Emre Sonmez, Eric Wu, Kevin Wu, Pankaj K. Agarwal, Jun Yang (Duke University)
Naeemul Hassan, Afroza Sultana, Gensheng Zhang, Chengkai Li (University of Texas at Arlington)
Cong Yu (Google Research)

1 Introduction

"Big data" have arrived. The increasing abundance of data brings many opportunities for journalism, from discovering interesting stories to fact-checking claims. Unfortunately, the skill of making sense out of data is in short supply; software tools suitable for non-technical users are sorely lacking. The gap between the abundance of data and the shortage of human expertise seems to be widening. Unless we find a way to close this gap, big data will not achieve its potential for journalism. Moreover, the public may become more susceptible to "lies, d—ed lies, and statistics" that nitpick data to advance their own arguments.

The following examples illustrate the use of data in finding stories and checking facts, as well as the associated challenges.

Example 1 (Finding and Monitoring Interesting Claims)**.** *Attention-seizing claims backed up by data can benefit reporting in many ways. When used in stories, these "factlets" help make stories more concrete and memorable. Such factlets can also serve as leads that help reporters identify newsworthy stories. While factlets take many different forms, we consider the following three types commonly used in the media across a variety of domains, such as sports, weather, technology, crime, and finance.*

- Situational facts *[10], e.g.: "The social world's most viral photo ever generated 3.5 million likes, 170,000 comments and 460,000 shares by Wednesday afternoon."* (`cnbc.com/id/49728455`) *A "situational fact" is about an object* not *being less impressive than any other object within a certain context (e.g., all photos posted to Facebook) when they are compared by several measures (e.g., numbers of "likes", "comments" and "shares").*
- One-of-the-few claims *[12], e.g.: "Victor Oladipo scored 30 points and handed out 14 assists... only three other rookies have recorded at least 30 points and 14 assists in a game..."* (`espn.go.com/espn/elias?date=20140222`) *This statement is about an entity that is only dominated by at most three other entities.*
- Prominent streak factlets *[9, 15], e.g.: "This month the Chinese capital has experienced 10 days with a maximum temperature in around 35 degrees Celsius—the most for the month of July in a decade."* (`chinadaily.com.cn/china/2010-07/27/content_11055675.htm`) *A "prominent streak" is a long consecutive subsequence (e.g., 10 days of temperature) consisting of only large (small) values (e.g., all above a value close to 35 degrees) within a sequence of values (e.g., daily maximum temperature of Beijing).*

Methods for ***finding factlets automatically from data*** *are crucial in reducing the burden posed by ever-growing data given limited reporting resources. Today, discovery of interesting claims in news relies largely on intuition and effort of domain experts. An expert, impressed by an event such as outstanding performance of a player in a game, would hypothesize a factlet and then manually craft a database query to check it. This approach appears to the one taken by Elias Sports Bureau in supplying factlets to sports media [3]. However, manual discovery is laborious and error-prone. It leads to poor coverage of local or non-mainstream events, and it ties up precious human expertise that could be otherwise devoted to more important journalistic activities.*

Another challenge is ***automatic, real-time monitoring of factlets*** *from data that come fast. The value of a news piece diminishes rapidly after an event takes place. For example, sports reporters need to present sensational factlets quickly as they emerge during a game; delays make fans less interested in the records and risk losing them to rival media. For another example, to help investors make timely decisions, financial reporters must be able to identify and present new factlets at the speed of the market.*

Example 2 (Fact-Checking Claims in Political Campaign Ads)**.** *This example is straight 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."*[1] *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.*

Several challenges are evident from this example. First, ***the claim may be vague****. In this example, the claim does not specify the exact target and period of comparison. The reason for omitting such details might be to make the claim short and memorable, but in some cases vagueness can be a way of hiding the fact that the claim cherry-picked data. Currently, seeking clarification to vague claims is mostly a manual process that can significantly hold up fact-checking.*

[1]This ad was in response to an earlier ad attacking Marshall for voting with Nancy Pelosi "almost 90 percent of the time," which, not surprisingly, also tailored the claim in ways to further its own argument.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

*The second challenge is that fact-checking goes beyond checking correctness—**a correct claim can still mislead**. For example, as we have seen in the discussion above, the claim about Marshall's voting record is correct (once we resolve the ambiguity), but it fails to present the overall picture fairly—it uses a period of comparison that exaggerates the percentage of agreement, and it does not put this percentage in a proper context (that the percentage of agreement between Republicans and Democrats is actually not low when we consider all votes including non-controversial ones).*

*Finally, an important part of a fact-checker's job is to convey the conclusion of fact-checking effectively to an audience. A proven strategy is to **construct short counterarguments** that are derived from the same data but lead to conclusions that differ from the original claim. For example,* `factcheck.org` *uses two counterarguments against the original claim on Marshall's voting record—one that shows a lower percentage of agreement when the period of comparison is changed, and another that shows a comparable percentage when substituting Marshall with somebody (Clyburn) who represents Democrats' voting pattern. Constructing effective counterarguments from data requires expertise and currently demands a lot of manual effort.*

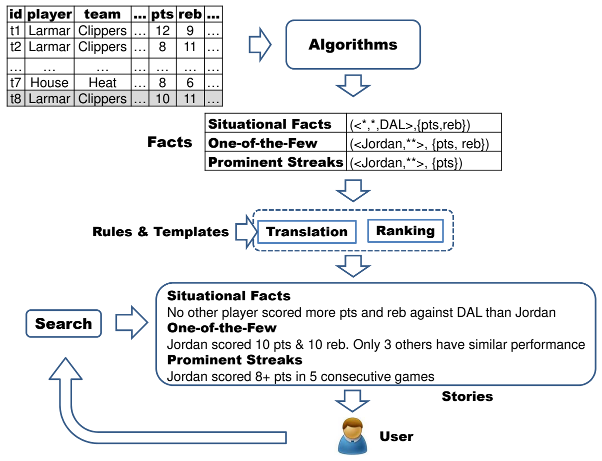
Today, these challenges are limiting the scale and scope of fact-checking despite availability of data. Imagine if we could automate the fact-checking process such that we can disambiguate vague claims, assess the quality of claims beyond correctness, and come up with counterarguments automatically using data. Journalists would be able to check more claims faster and more accurately, allowing them to expand fact-checking to levels previously unattainable with purely manual efforts.

We have developed a suite of computational techniques for finding and checking claims based on structured data [13, 9, 12, 10, 15]. Our techniques are able to capture human intuition of what constitutes "good" (not just correct) claims, and allows us to specify relevant domain knowledge and formulate various journalistic tasks—such as finding interesting claims from data, monitoring evolving data for such claims, disambiguating vague claims, and constructing counterarguments—as computational problems that can be solved automatically. Our framework is surprisingly versatile: it is able to capture various notions of claim quality, and can be applied to many claim types across different domains.

This demonstration showcases two systems, ***uClaim*** and ***iCheck***. uClaim focuses on finding interesting claims from data. It automatically maintains the set of interesting claims, in real time, as new data become available. iCheck focuses on checking claims based on data. It can automatically "reverse-engineer" vague claims to remove ambiguity, and come up with counterarguments to low-quality claims. Both systems include visualization and social features that help users understand the claims and share their insights. Earlier versions of these systems have been demonstrated in other venues [7, 14].

2 uClaim: Finding and Monitoring Claims

The uClaim system automates finding and monitoring of the three types of facts discussed in Example 1. uClaim integrates the various algorithms in [9, 12, 10, 15]. It incorporates all three types of facts into a unified suite of data model, algorithm framework and fact ranking measure. Figure 1 illustrates the components of uClaim. Given an ever-growing database table, upon the arrival of a new tuple t into the table, uClaim checks if t triggers any new situational facts, one-of-the-few facts, and prominent streaks. It also ranks these facts together by several different ranking measures. Furthermore, uClaim provides multiple features in striving for an



id	player	team	...	pts	reb	...
t1	Larmar	Clippers	...	12	9	...
t2	Larmar	Clippers	...	8	11	...
...	...	...	...	...	...	...
t7	House	Heat	...	8	6	...
t8	Larmar	Clippers	...	10	11	...

Situational Facts	(<*,*,DAL>,{pts,reb})
One-of-the-Few	(<Jordan,**>, {pts, reb})
Prominent Streaks	(<Jordan,**>, {pts})

Situational Facts
No other player scored more pts and reb against DAL than Jordan
One-of-the-Few
Jordan scored 10 pts & 10 reb. Only 3 others have similar performance
Prominent Streaks
Jordan scored 8+ pts in 5 consecutive games

Figure 1: uClaim System Architecture

end-to-end system. By using rules and templates, the discovered facts are translated into textual news leads and presented to users; it provides a faceted interface for browsing the discovered facts; it supports keyword-based search of facts; it also allows checking the profiles of entities and comparing the entities based on the facts triggered by them.

Figure 2 shows uClaim's customized GUI for NBA (National Basketball Association) data, where tuples—players' statistics in individual games—are updated during and after the games. The GUI's structure is dataset-agnostic and can be adapted to data from other domains. We will demonstrate uClaim on an NBA dataset and a weather dataset. The NBA dataset has 317,371 tuples of NBA box scores from 1991-2004, on 8 dimension attributes (e.g., *player*, *team*, *year*, etc.) and 7 measure attributes (e.g., *points*, *rebounds*, etc.) The weather dataset has 7.8 million daily weather forecast records for 5,365 locations of UK from Dec. 2011 to Nov. 2012. It has 7 dimension attributes and 7 measure attributes.

The GUI consists of four areas, as follows.

1. Stories This area shows a ranked list of textual news leads (stories) translated from facts that have ever been discovered and are still valid. It also allows users search for translated stories by keywords. The translation is guided by a set of templates and rules. If the "More Like This" button beside a story is clicked, uClaim shows similar stories in a pop-up window (top-right corner of Figure 2). The pop-up window also presents bar charts to compare the stories by their values on measure attributes.

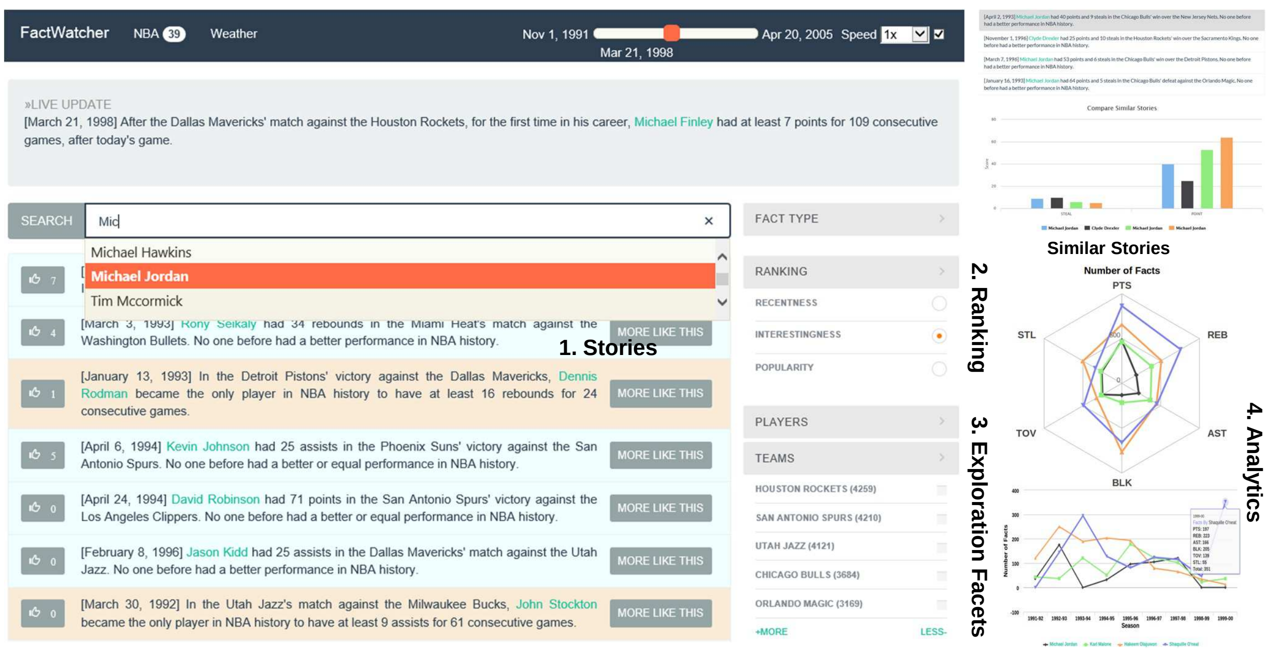
2. Ranking uClaim allows users to choose from several ways of ranking facts and their corresponding stories.

Interestingness This default option ranks facts by significance of measure attribute values in the facts.

Popularity This option ranks facts by numbers of times users click the thumb-up buttons beside the stories.

Recentness This option simply orders facts by their triggering tuples' timestamps.

3. Exploration This area presents a faceted interface for exploring the stories. Each facet corresponds to a dimension or measure attribute. Under the facet for a dimension attribute, the attribute values are associated with and ordered by numbers, which indicate how many facts involve the values. For instance, Figure 2 shows that there are 4,259 facts related to *team*=*Houston Rockets*. uClaim places a checkbox beside each attribute value. A user can select/unselect the checkboxes across multiple facets. The selected values within one facet correspond to a disjunctive condition, and the disjuncts from different facets form a conjunctive condition.



»LIVE UPDATE
[March 21, 1998] After the Dallas Mavericks' match against the Houston Rockets, for the first time in his career, Michael Finley had at least 7 points for 109 consecutive games, after today's game.

[March 3, 1993] Rony Seikaly had 34 rebounds in the Miami Heat's match against the Washington Bullets. No one before had a better performance in NBA history.

[January 13, 1993] In the Detroit Pistons' victory against the Dallas Mavericks, Dennis Rodman became the only player in NBA history to have at least 16 rebounds for 24 consecutive games.

[April 6, 1994] Kevin Johnson had 25 assists in the Phoenix Suns' victory against the San Antonio Spurs. No one before had a better or equal performance in NBA history.

[April 24, 1994] David Robinson had 71 points in the San Antonio Spurs' victory against the Los Angeles Clippers. No one before had a better or equal performance in NBA history.

[February 8, 1996] Jason Kidd had 25 assists in the Dallas Mavericks' match against the Utah Jazz. No one before had a better performance in NBA history.

[March 30, 1992] In the Utah Jazz's match against the Milwaukee Bucks, John Stockton became the only player in NBA history to have at least 9 assists for 61 consecutive games.

[April 2, 1993] Michael Jordan had 40 points and 9 steals in the Chicago Bulls' win over the New Jersey Nets. No one before had a better performance in NBA history.

[November 1, 1996] Clyde Drexler had 25 points and 10 steals in the Houston Rockets' win over the Sacramento Kings. No one before had a better performance in NBA history.

[March 7, 1996] Michael Jordan had 53 points and 6 steals in the Chicago Bulls' win over the Detroit Pistons. No one before had a better performance in NBA history.

[January 16, 1993] Michael Jordan had 64 points and 5 steals in the Chicago Bulls' defeat against the Orlando Magic. No one before had a better performance in NBA history.

Figure 2: uClaim User Interface

They together correspond to multiple constraints. Each fact (story) displayed in the "stories" area must satisfy one such constraint.

Beside the facet for a measure attribute, uClaim presents a slidebar and a button (not shown in Figure 2). A user can click the button to enable/disable a measure attribute. Accordingly the "stories" area displays stories related to one or more enabled measure attributes. The user can also use the slidebar to set minimum and maximum desired values on an enabled attribute. Accordingly the displayed facts must have values on that attribute within the range.

4. Analytics This area visualizes the statistics on facts related to entities selected by a user. The "stories" area highlights the entities in stories, e.g., *Jason Kidd* and *David Robinson* in Figure 2. When a user clicks on an entity, it is added to the entity list in the "analytics" area. The user can remove an entity by clicking the crossmark beside it. The bottom part of the "analytics" area is a line chart, which shows one line per selected entity that represents the number of facts (among the displayed facts in the "stories" area) triggered by the entity over each time period (e.g., each NBA season). When the user hovers the mouse on a data point, a pop-up box shows, for each measure attribute, the number of facts related to the measure attribute. The top part of this area is a radar chart, which shows one polygon per selected entity that represents how many facts triggered by the entity are related to each measure attribute.

Computational Challenges The algorithmic problem that uClaim solves is finding constraint-measure pairs that qualify a new tuple t as a fact. A straightforward brute-force approach would compare t with every historical tuple to determine if t stands out, repeatedly for every relevant constraint-measure pair. The obvious low-efficiency of this approach has three culprits—exhaustive comparison with *every tuple*, under *every constraint*, and over *every combination of measure attributes*. [9, 12, 10, 15] presented efficient algorithms to counter these issues. For example, in [10] the algorithm finds and monitors situational facts by pruning from consideration clearly weak tuples, constraints, and measures. The key to the algorithm is to substantially reduce the search space of possible facts while at the same time make sure all valid facts are found. For a dataset with modest size (such as the aforementioned NBA data), the brute-force approach would take months to finish, while the developed algorithm only requires hours.

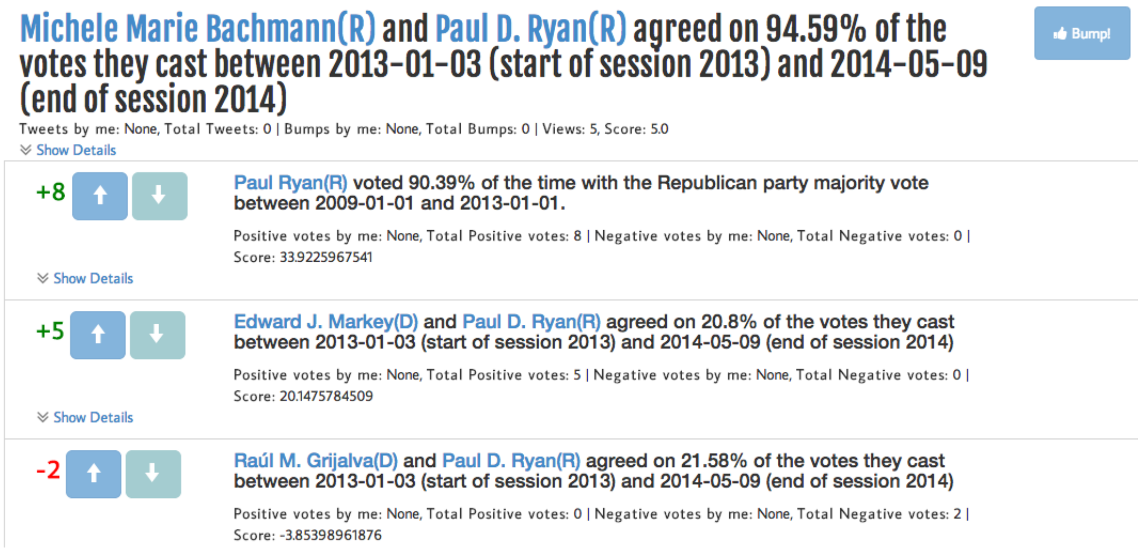
With regard to prominent streaks, the solution in [9, 15] hinges upon the idea of separating two steps—*candidate streak generation* which generates a small number of candidate streaks ending at the new tuple without exhaustively considering all possible streaks, and *skyline operation* which maintains a dynamic set of prominent streaks by performing dominance comparison between existing prominent streaks and candidate streaks. A brute-force approach to candidate streak generation would enumerate all $\frac{n(n+1)}{2}$ possible streaks in an n-tuple sequence as candidates. On the contrary, the algorithm in [9, 15] only needs to produce at most n candidate streaks.

An additional challenge is to make the algorithms *incremental* in order to monitor claims. Instead of redoing everything from scratch whenever a new database tuple comes, our algorithms are capable of leveraging the results computed for previous tuples and only incurring computational cost for the delta due to the latest tuple.

3 iCheck: Checking Claims

The iCheck system automatically "checks" claims based on structured data. This demonstration of iCheck focuses on two types of claims from two domains: "one-of-the-few" claims about players in Major League Baseball (similar in form to the one described in Example 1), and vote correlation claims about United State legislators (similar to the one described in Example 2). A user can ask iCheck to analyze a claim of either type. If this claim is vague, iCheck will first show a list of precisely stated claims representing the most probable ways of resolving the ambiguity. Once the user selects a precisely stated claim for analysis, iCheck will present a list of claims supporting or countering it. From this list, the user can get an intuitive sense of how strong or misleading the claim is.

The iCheck demonstration features data from two domains: Major League Baseball and congressional voting records. Major League Baseball dataset consists of season statistics of teams, pitchers, bat-


Michele Marie Bachmann(R) and Paul D. Ryan(R) agreed on 94.59% of the votes they cast between 2013-01-03 (start of session 2013) and 2014-05-09 (end of session 2014)

Tweets by me: None, Total Tweets: 0 | Bumps by me: None, Total Bumps: 0 | Views: 5, Score: 5.0

Show Details

+8 Paul Ryan(R) voted 90.39% of the time with the Republican party majority vote between 2009-01-01 and 2013-01-01.

Positive votes by me: None, Total Positive votes: 8 | Negative votes by me: None, Total Negative votes: 0 | Score: 33.9225967541

Show Details

+5 Edward J. Markey(D) and Paul D. Ryan(R) agreed on 20.8% of the votes they cast between 2013-01-03 (start of session 2013) and 2014-05-09 (end of session 2014)

Positive votes by me: None, Total Positive votes: 5 | Negative votes by me: None, Total Negative votes: 0 | Score: 20.1475784509

Show Details

-2 Raúl M. Grijalva(D) and Paul D. Ryan(R) agreed on 21.58% of the votes they cast between 2013-01-03 (start of session 2013) and 2014-05-09 (end of session 2014)

Positive votes by me: None, Total Positive votes: 0 | Negative votes by me: None, Total Negative votes: 2 | Score: -3.85398961876

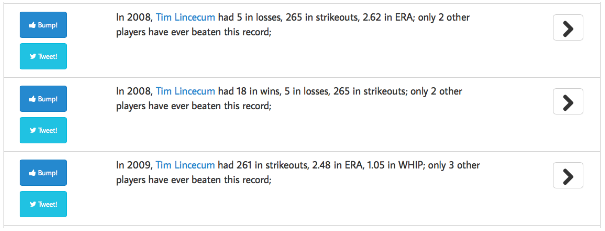
Figure 3: iCheck claim centric political view

ters, and fielding dating back to the start of the league. Congressional voting records, in brief, consist of representatives voting on bills. A bill can have many votes as it propagates through Congress, and there are many different bill types and designations.

The iCheck interface provides *entity-centric* and *claim-centric* views to users. The entity-centric view allows a user to browse the list of claims known to iCheck about an entity (be it a player in Major League Baseball or a US legislator). These claims include those entered by users as well as those generated automatically by iCheck itself (as in uClaim). These claims can be sorted by popularity (based on the level of user activities on each claim) and quality (based on either objective measures or votes by users), allowing the user to spot interesting claims easily. From the entity-centric view, the user can also enter a claim about an entity; iCheck will disambiguate the newly entered claim if necessary, and compare it with known claims to avoid duplication.

The claim-centric view allows a specific claim to be scrutinized. For a precisely stated claim, this view lists the counterarguments that iCheck generates. For a vague claim, this view lists the precisely stated claims that iCheck reverse-engineers from the vague claim. On the claim-centric view, users can comment on the claim, and vote on which counterarguments are the most effective, or which reverse-engineered claims are the most probable precise versions of vague claims.

For example, Figure 3 shows the iCheck claim-centric view for a political vote-correlation claim. Arguments can be voted on and sorted based on relative quality of their arguments, and can be supportive or counter to the original claim. Figure 4 shows the entity-centric view in the MLB domain. In this domain, a player generates 'one of the few' claims.


In 2008, Tim Lincecum had 5 in losses, 265 in strikeouts, 2.62 in ERA; only 2 other players have ever beaten this record;

In 2008, Tim Lincecum had 18 in wins, 5 in losses, 265 in strikeouts; only 2 other players have ever beaten this record;

In 2009, Tim Lincecum had 261 in strikeouts, 2.48 in ERA, 1.05 in WHIP; only 3 other players have ever beaten this record;

Figure 4: iCheck MLB player centric view

Computational Challenges The key insight behind our approach to fact-checking is that we tweak the view in which a claim presents underlying data, and then observe how its conclusion changes. To this end, we treat a claim as parameterized query over a database; the parameters of the query correspond to how the claim "picks" its view of data, and the answer of the query correspond to the conclusion of the claim. For example, the vote correlation claim in Example 2 can be considered as a query that computes the similarity between two time series of votes over a given time interval, over the database of Congressional votes. The query has four parameters: the two legislators being compared, and the starting and end times of the comparison period. The original claim is represented by a particular combination of settings for these four parameters. By perturbing the settings of these four parameters, we get many insights. If slight changes to the period of comparison cause the percentage of vote agreement between Marshall and Boehner to drop significantly, we know that the original claim is not robust. If we replace Marshall with other well-known Democrats and observe that the percentage of vote agreement with Boehner remains high, we know that percentage in the original claim is not as high as

Technically, we model all possible perturbations of parameters in a query as a *parameter space*, and the variation of query answer over this space as a *query response surface*. Over the parameter space, we use a *relative claim strength function* to measure how much a perturbed claim strengthens/weakens the conclusion of the original claim, and a *relative claim sensibility function* to measure how well a perturbed claim captures the context of the original claim. One way to think of the sensibility function is to view it as a pdf/pmf on the parameter space. Using these two functions, we can define quantitative measures for various properties of a claim as aggregated properties of the query response surface. We can also model the tasks of reverse-engineering vague claims and finding counterarguments as optimization problems over the query response surface. For example, given a vague claim with a conclusion and a rough guess of its parameter settings, we can attempt to reverse-engineer it by searching for high-sensibility parameter settings that lead to a query result matching that of the vague claim.

However, these problems are not computationally trivial tasks. A brute-force approach has to evaluate the query for every possible parameter setting. For example, for the vote correlation claim, the number of possible parameter settings is quadratic in the number of legislators and in the number of votes. Even if we perturb only one out of the four parameters (one of the legislator being compared), it would take about 5 seconds to consider all possibilities, which would not satisfy the needs for real-time analysis. However, by exploiting the properties of the parameter sensibility function to prioritize the search, we are able to find satisfactory solutions under one second. For the general case, we have developed an algorithmic framework that enables efficient instantiations of "meta" algorithms by supplying appropriate algorithmic building blocks, such as prioritized search and group evaluation of multiple perturbations; for details, see [13].

4 Conclusion

Related Work The problem of finding interesting claims from data is broadly related to the field of *data mining* [11, 6]. Our framework allows flexible mining tasks to be specified using query templates (e.g., in SQL). Although data mining, machine learning, and statistics offer many alternative approaches towards finding patterns in data, our approach has the appealing feature that the discovered patterns readily translate to claims that are easy to understand by a lay person. In contrast, results from more sophisticated approaches (say, SVM classification) are harder to explain and cannot be directly cited in stories.

Companies such as *Narrative Science* and *Automated Insights* are able to automatically generate news stories based on data for domains such as business and sports, thereby alleviating humans from writing narratives that are largely template-based. We do not seek to generate complete stories; instead, we focus on supplying reporters with interesting leads and factlets that can be used in full stories. Thus, our work can complement what these companies offer. Other companies such as *Elias Sports Bureau* provides data as well as statistics and factlets derived from data. They usually focus on particular domains and provide their service for a fee. We have a more general framework for automation, and additionally support fact-checking.

Organizations such as `factcheck.org` and `PolitiFact.com` rely on their expert editorial staff to check claims. However, manual approaches are costly to scale because of the demand on human expertise and effort. *FactMinder* [5] 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* [4] 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.

Using *question answering* systems, such as IBM's *Watson* and *WolframAlpha*, and *natural language querying* systems, we can check the correctness of some claims in natural language. However, these systems are not yet capable of handling claims that correspond to complex queries. Furthermore, they can be used to check only the correctness of claims, but as we have argued earlier, fact-checking goes beyond checking correctness.

Future Work Our work to date is only a first step toward a computational approach to fact-checking. We have not addressed a number of related issues, such as automatically mapping natural-language claims to known templates, identifying and extracting data from relevant sources, and assessing the quality of these sources.

Furthermore, finding stories and checking facts *in general* are very challenging. Not all leads come from data. Not all lies can be detected by inspecting data and numbers alone (see, e.g., [8, 1, 2]). 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.

References

[1] Joel Best. *Damned Lies and Statistics: Untangling Numbers from the Media, Politicians, and Activists*. University of California Press, 1 edition, May 2001.

[2] Joel Best. *More Damned Lies and Statistics: How Numbers Confuse Public Issues*. University of California Press, 1 edition, September 2004.

[3] Neil Best. http://www.newsday.com/sports/columnists/neil-best/hirdt-enjoying-long-run-as-stats-guru-1.3174737, Sept. 2011. Accessed: Jul. 2013.

[4] Rob Ennals, Beth Trushkowsky, and John Mark Agosta. Highlighting disputed claims on the Web. In *Proceedings of the 2010 International Conference on World Wide Web*, pages 341–350, Raleigh, North Carolina, USA, April 2010.

[5] François Goasdoué, Konstantinos Karanasos, Yannis Katsis, Julien Leblay, Ioana Manolescu, and Stamatis Zampetakis. Fact checking and analyzing the Web. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 997–1000, New York City, New York, USA, June 2013.

[6] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3 edition, 2005.

[7] Naeemul Hassan, Afroza Sultana, You Wu, Gensheng Zhang, Chengkai Li, Jun Yang, and Cong Yu. Data in, fact out: Automated monitoring of facts by FactWatcher. *Proceedings of the VLDB Endowment*, 7(13), 2014.

[8] Darrell Huff. *How to Lie with Statistics*. W. W. Norton & Company, reissue edition, 1993.

[9] Xiao Jiang, Chengkai Li, Ping Luo, Min Wang, and Yong Yu. Prominent streak discovery in sequence data. In *Proceedings of the 2011 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1280–1288, San Diego, California, USA, August 2011.

[10] Afroza Sultana, Naeemul Hassan, Chengkai Li, Jun Yang, and Cong Yu. Incremental discovery of prominent situational facts. In *Proceedings of the 2014 International Conference on Data Engineering*, Chicago, Illinois, USA, March 2014.

[11] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[12] You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. On "one of the few" objects. In *Proceedings of the 2012 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495, Beijing, China, August 2012.

[13] You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. Toward computational fact-checking. *Proceedings of the VLDB Endowment*, 7(7):589–600, 2014.

[14] You Wu, Brett Walenz, Peggy Li, Andrew Shim, Emre Sonmez, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. iCheck: Computationally combating "lies, d—ned lies, and statistics". In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, Snowbird, Utah, USA, June 2014.

[15] Gensheng Zhang, Xiao Jiang, Ping Luo, Min Wang, and Chengkai Li. Discovering general prominent streaks in sequence data. *ACM TKDD*, 2014.