Scrutinizer: Fact Checking Statistical Claims

Georgios Karagiannis[†*] Mohammed Saeed[‡*] Paolo Papotti[‡] Immanuel Trummer[†]
[†]Cornell University, USA [‡]Eurecom, France
{ gk446,lt224 }@cornell.edu, { papotti,mohammed.saeed }@eurecom.fr

ABSTRACT

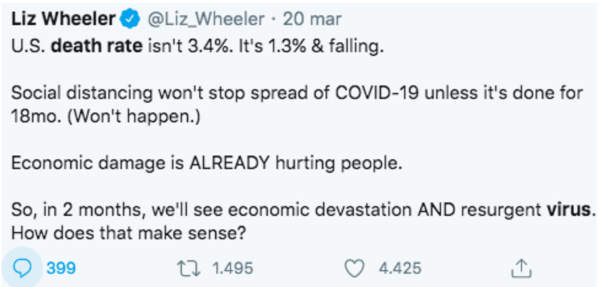
We demonstrate Scrutinizer, a system that supports human fact checkers in translating text claims into SQL queries on an associated database. Scrutinizer coordinates teams of human fact checkers and reduces their verification time by proposing queries or query fragments over relevant data. Those proposals are based on claim text classifiers, that gradually improve during the verification of multiple claims. In addition, Scrutinizer uses tentative execution of query candidates to narrow down the set of alternatives. The verification process is controlled by a cost-based optimizer that plans effective question sequences to verify specific claims, and prioritizes claims for verification. In this demonstration, we first show how our system can assist users in verifying statistical claims. We then let users come up with new, unseen claims and show how the system effectively learns new queries with little user feedback.

PVLDB Reference Format:
Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, Immanuel Trummer. Scrutinizer: Fact Checking Statistical Claims. *PVLDB*, 13(12): xxxx-yyyy, 2020.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

Data is often disseminated in the form of textual claims and reports, summarizing important statistics. For authors of such documents, it is time-consuming and tedious to ensure the correctness of each single claim. Nevertheless, erroneous claims about data are not acceptable in many scenarios as each mistake can have dire consequences. Those consequences reach from embarrassing retractions (in case of scientific papers [3]) to legal or financial implications (in case of business or health reports [1]). We demonstrate SCRUTINIZER, a system that helps teams of fact checkers to verify consistency of text w.r.t. data faster.

The SCRUTINIZER system has been developed in collaboration with the International Energy Agency (IEA). This NGO regularly publishes scientific reports encompassing hundreds of pages, requiring months of verification efforts by internal teams of experts. Using real reports and data, we will demonstrate to visitors how SCRUTINIZER reduces verification overheads in this scenario. More recently, we have published an online version of our system that verifies single statistical claims about the spread and effects of the coronavirus[1]. This version has attracted thousands of users and has been covered in national newspapers[2] – we will use it for our demonstration as well.



Liz Wheeler @Liz_Wheeler · 20 mar
U.S. **death rate** isn't 3.4%. It's 1.3% & falling.

Social distancing won't stop spread of COVID-19 unless it's done for 18mo. (Won't happen.)

Economic damage is ALREADY hurting people.

So, in 2 months, we'll see economic devastation AND resurgent **virus**. How does that make sense?

Figure 1: A Tweet with an incorrect claim about the death rate of the coronavirus disease.

EXAMPLE 1. *Consider the claim in the Tweet in Figure 1. There are multiple online sources of official data for the virus outbreak that can be used to demonstrate that it is incorrect. However, a content moderator would have to find the relevant dataset and manually write a query over such data to collect the relevant information. In the example:*

```
SELECT b.March/a.March
FROM totalCases a, totalDeaths b
WHERE a.Country = 'USA', b.Country = 'USA'
```

Finally, the expert compares the output of the query with the claim and can eventually flag the content as incorrect.

Gathering data for the claim at hand and composing the right query for the validation takes expertise over the domain and data skills, typically taking several minutes for a single claim. To reduce this time, given a document with statistical claims and related datasets, our system helps users to translate claims into corresponding SQL queries, to verify and to potentially correct them. Doing so entails the following challenges. **Text analysis:** Converting a textual claim

[*] The first two authors contributed equally.

[1] https://coronacheck.eurecom.fr
[2] https://bit.ly/39BTJCE

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

to a query is difficult because claims are expressed in natural language, do not use a fixed vocabulary, and come from multiple authors with different wording and style. **Query complexity:** Our analysis of thousands of claims with their corresponding queries reveals that the queries associated with claims are diverse, going from simple selections to mathematical operations involving grouping of values, aggregations, and functions. **Large corpus of datasets:** Given a corpus of datasets, it is not clear which one(s) should be used to verify a new statistical claim. For instance, when verifying IEA reports, the correct data set must be selected from hundreds of alternatives.

We tackle the above challenges with a novel system, Scrutinizer [5], that analyzes claims via three primary methods: machine learning (ML) and natural language processing (NLP) for analyzing claim text, feedback from human domain experts for validating candidate queries, and query generation and tentative execution based on a large library of functions. The interpretable SQL queries are finally exposed to users so that they can either validate or flag as false the claim at hand.

Prior verification systems [4] assume that a single user verifies a short document based on a single data source. Scrutinizer is targeted at the verification of many, complex claims by teams of fact checkers. For instance, Scrutinizer learns to recognize new types of claims and queries as more claims from a given domain are verified. It supports claim queries connecting multiple data sources or containing complex, arithmetic expressions. Also, it uses a cost-based optimizer to plan the verification of large documents in order to minimize manual overheads. As outlined in more detail in our full paper [5], those features turn out to be crucial in the scenarios we are aiming for.

The demonstration will convey the following four primary insights about the Scrutinizer system.

1. We introduce ML models that extract the fragments of the final SQL query from the textual claim. We will show how classifiers identify datasets, attributes, rows and mathematical operations that verify the claim, even for claim expressed in very different wordings

2. For any claim verification, we will explain how the decision has been taken by the system. Such explanations are expressed as declarative queries over relational data. The explanations are produced by a query generation algorithm combining the information extracted from the text.

3. For large documents with many claims, we show how a scheduling algorithm for planning both the sequence of claims to verify and the questions to ask to domain experts can minimize the verification cost.

4. We will verify statistical claims from energy reports and from coronavirus tweets to demonstrate that effective claim verification can be done in few seconds with pre-trained models.

2. SYSTEM OVERVIEW

Figure 2 shows an overview of Scrutinizer. The input consists of a text document, containing one or more claims, and a set of relations. If a database of previously checked claims is available, our system uses it for bootstrapping. If no such database is available, we use an active learning algorithm to steer the users in its creation. The output of the system is a verification report, mapping verified claims to queries while pointing out mistakes and potential updates to the text.

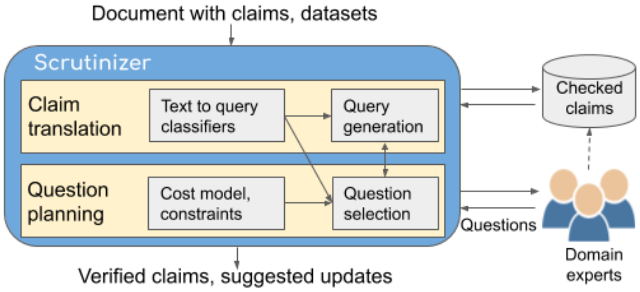


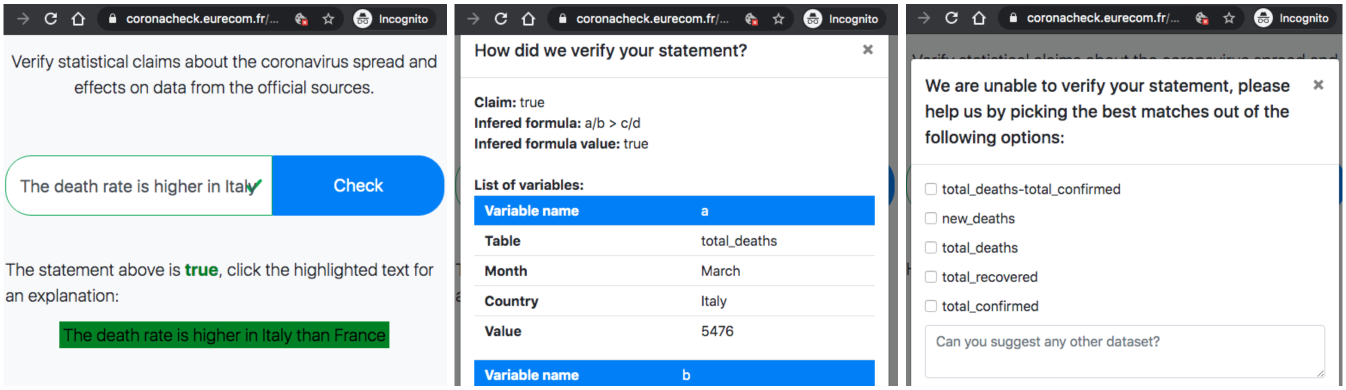
Figure 2: Architecture of Scrutinizer.

The system encompasses two primary components. The automated translation component leverages machine learning to identify the elements that define every claim, i.e., candidates for datasets, attributes, rows, and comparison operations. The question planning component interacts with human domain experts to verify such elements and the checking results, optimizing verification tasks for maximal benefit.

Algorithm 1 Main verification algorithm.

```
// Verify claims $C$ in text $T$ using models $M$
// and return verification results.
**function** Verify($T, C, M$)
    // Initialize verification result
    $A \leftarrow \emptyset$
    // While unverified claims left
    **while** $C \neq \emptyset$ **do**
        // Select next claims to verify
        $N \leftarrow$OptBatch($T, C, M$)
        // Select optimal question sequence
        $S \leftarrow$OptQuestions($N, M$)
        // Get answers from fact checkers
        $W \leftarrow$GetAnswers($N, M, S$)
        // Generate queries and validate claims
        $R \leftarrow$Validate($W$)
        $A \leftarrow W \cup R$
        // Remove answered claims
        $C \leftarrow C \backslash$Unanimous($N, A$)
        // Retrain text classifiers
        $M \leftarrow$Retrain($N, A$)
    **end while**
    // Return verification results
    **return** $A$
**end function**
```

Algorithm 1 describes the main steps in our workflow. Given claims C in a text document T and ML models M, claims are verified in batches by a team of human fact checkers. In each step, the algorithm selects an optimal batch N of claims for verification. Claim batches are selected based on multiple criteria, including expected verification overheads as well as their estimated utility for improving accu-


coronacheck.eurecom.fr/...
Incognito
coronacheck.eurecom.fr/...
Incognito
coronacheck.eurecom.fr/...
Incognito
Check
×
×

Verify statistical claims about the coronavirus spread and effects on data from the official sources.

The death rate is higher in Italy ✓

The statement above is **true**, click the highlighted text for an explanation:

The death rate is higher in Italy than France

How did we verify your statement?

Claim: true
Infered formula: a/b > c/d
Infered formula value: true

List of variables:

Variable name	a
Table	total_deaths
Month	March
Country	Italy
Value	5476
Variable name	b

We are unable to verify your statement, please help us by picking the best matches out of the following options:

□ total_deaths-total_confirmed
□ new_deaths
□ total_deaths
□ total_recovered
□ total_confirmed

Can you suggest any other dataset?

Figure 3: Screenshots of the demo for the check of a single claim (left) and its explanation (center). On the right, example of the feedback questions the system asks when model predictions have low confidence.

racy of the classifiers. For each selected claim in the current batch, we determine an optimal sequence of questions for the human checkers, minimizing expected verification time. Claims are validated or marked as erroneous, based on query evaluations. We remove the claims for which a verification result (i.e., either a verifying query or a decision that the claim is erroneous) can be calculated with sufficiently high confidence. Finally, the classifiers are retrained, based on the newly obtained classification results.

We detail the two main components in the following.

2.1 Text to Query Translation

For a given textual claim, the system starts by executing four classifiers over it. In the case of documents as input, we identify the text relevant for the statistical claim with existing tools for this task [4]. Given the claim, the classifiers identify four elements that are key for the query generation process and claim verification. The first three are basic elements of every query: relevant relations, primary keys values (rows), and attributes names. The fourth classifier is in charge of identifying a generic formula with variables in the place of keys and attribute values. This formula gets instantiated on the dataset at hand and becomes the combination of functions in the SELECT clause. If an element cannot be predicted with high confidence, the system asks for user input to build the query.

Example 2. *Consider again the (false) claim "U.S. death rate is **1.3%** in March 18". The first classifier identifies that* Deaths *and* ReportedCases *relations can be used to verify it; the second classifier recognizes that rows reporting values for* U.S.A. *should be used; the third classifier returns* March 18 *as the attribute of interest, and, finally, the fourth classifier returns the formula $\frac{a}{b}$. The output of the query is then compared to value 0.013 (1.3%) to assess the claim.*

To get good accuracy results in the prediction, we resort to active learning. This is useful for cases in which previously checked claims are immediately used to derive training data for the classifiers, but also enables the use of our system for cases where previous checks are not available. Previous claim checks are also important for generalizing the specific functions used in the past verification into generic formulas with variables. This step enables us to (i) reuse formulas on unseen claims and (ii) have a number of classes (for the prediction) as small as possible.

As we cannot assume that the first prediction is always the correct one in practice, in cases with low confidence we resort to users validating the relations, rows, and attributes predictions. Once we have this "context" information, we predict the top formulas with the last classifier and generate all the possible queries that combine context and formulas. The complexity raised by this combination is in the assignment of the elements of the query to the variables in the formula. Consider two attributes A_1 and A_2 identified for a certain row and a formula of the form "$a - b$", the system does not know if A_1 is assigned to variable a or b.

Example 3. *Consider the claim "The death rate is higher in Italy than France.", as reported in the left hand side of Figure 3 with a screenshot of the GUI. Given the predictions for relations (*totalDeaths, totalCases*), rows (*Italy,France*), attributes (*March*) and formula $a/b > c/d$, the query generator produces all the possible bindings for variables $a-d$ over the relation, for Italy and France rows, and with attribute for the current month. In one assignment, a is correctly bound to a row in relation* totalDeaths, *with key value* Italy *and attribute* March, *while in a different assignment a is bound to the same row and attribute but on relation* totalCases, *and so on. One of these queries is not empty, thus validating the claim. The query can be visualized as an explanation to the user, as reported in the middle screenshot in Figure 3.*

The assignment operation is done in a brute force fashion, but, thanks to the pruning power of the context, it is usually achieved in milliseconds.

2.2 Question Planning

Our system relies on human fact checkers to verify automatically generated translations of claims to queries. As soliciting feedback from human workers is expensive, the question planning component uses cost-based optimization to determine effective question sequences. Question planning consists of two sub-tasks. First, for a fixed claim with low confidence in the models' predictions, we choose a sequence of questions allowing us to verify that claim with minimal expected overhead. Each question either solicits users to verify generated query fragments, or to propose suitable query fragments themselves. Second, we need to decide the order in which claims are verified. When selecting claims to verify next, we take into account expected verification overheads as well as their value as training samples for our classifiers (used for automated claim verification).

Our search space for the first sub-task (determining optimal questions for single claims) are sequences of screens containing questions for the fact checkers. Each screen focuses on one property of the query representing the claim. For instance, a screen may focus on the data source that a claim refers to. Or it may focus on arithmetic expressions in the query's select clause. Each screen shows ranked answer options, generated by the classifiers, and enables workers to enter their own answers if the correct option is not recognized. The planning component decides how many options to show per screen, which screens to show, and in which order to show them. Its decisions are based on a cost model, taking into account dependencies between different query properties (e.g., having verified the data source excludes options with regards to query attributes) and estimating overheads for fact checkers when verifying given options or suggesting new ones.

Example 4. *Figure 3 (right) shows an example of a screen generated for a claim that could not be verified automatically with high confidence. Here, human workers are asked to select one out of multiple possible query aggregates. Depending on the amount of prior training data available, claims are typically verified by a sequence of such screens, focusing on different topics. Note that the screen shown in Figure 3 does not offer the option of entering new suggestions. This is only possible if properties of the data source and domain exclude alternative options.*

The second sub-task (prioritizing claims) is important for large documents containing multiple claims. For instance, reports by the IEA typically contain hundreds of claims that take weeks to verify. Each verified claim serves as training samples for our classifiers. Prioritizing claims with low classification confidence tends to improve classifier accuracy faster. On the other side, verifying such claims is often more expensive as human checkers cannot benefit from high-quality suggestions. Also, checkers prefer working through claims in document order as that avoids context switching overheads. We use cost-based optimization to strike a balance between those extremes. In selecting the next claims to verify, we consider expected verification cost (including cost for switching between different document parts) as well as expected utility for the classifiers. The resulting optimization task is formalized as an integer linear programming problem and answered by a corresponding solver.

3. DEMONSTRATION

We consider two scenarios for our demonstration. Those scenarios are motivated by real-world use cases in which our system has been applied. First, we consider the "2018 World Energy Outlook Report" by the IEA[3] with associated data. This report summarizes results of sophisticated models predicting future energy consumption on hundreds of pages. We are collaborating with the IEA to reduce fact checking overheads when verifying such reports via our system. Second, we demonstrate verification of claims related to the Coronavirus, using data sets published by organizations such as CDC (Center of Disease Control) and WHO (World Health Organization) as data sources [2]. This scenario is motivated by the spread of misinformation concerning the pandemic. An online version of our system, targeting such claims and available since March 2020, has already attracted more than twelve thousands of users.

We will prepare different demonstrations, targeted at visitors with different time budgets. First, visitors can get a quick impression of our system, without spending too much time, by applying it to single claims. Here, we will use the online version of our system for verifying claims about the coronavirus disease (COVID-19) outbreak. This version has already been trained for this domain, leveraging input by a large number of users as training data. Visitors can enter claims concerning the spread of the virus (e.g., "The death rate in Italy is much higher than in France." or "The total number of confirmed cases in USA remained constant from February to March"), and obtain a verification result as answer. Beyond the standard verification interface, we will show to interested visitors the queries into which claims are translated and will explain the translation process.

Second, for visitors with more time, we will prepare a demonstration putting them into the roles of professional fact checkers at IEA. We will prepare several extracts (one to two paragraphs per extract) from the 2018 World Energy Outlook report. Initially, we will give visitors the chance to experience the current default fact checking process. This process involves manually identifying suitable data sources (choosing from hundreds of alternatives) as well as translating claims into queries (involving complex arithmetic expressions) on these data. Afterwards, we will give visitors the opportunity to verify the same extracts using our system. Here, visitors benefit from suggestions for data sets and query properties that are in most cases correct. If desired, we will exploit a functionality of our system that times users during verification (used in our prior experiments) to give them a better impression of the efficiency gains.

Acknowledgements. We thank Davide Rossi D'Ambrosio and the IEA staff for the valuable feedback on the fact checking process and Youssef Doubli for his work on the CoronaCheck Web site.

4. REFERENCES

[1] J. S. Ash, M. Berg, and E. Coiera. Some unintended consequences of information technology in health care: the nature of patient care information system-related errors. *JAMIA*, 11(2):104–112, 2004.

[2] E. Dong, H. Du, and L. Gardner. An interactive web-based dashboard to track COVID-19 in real time. *Lancet Infect Dis*, 2020.

[3] M. Hosseini, M. Hilhorst, I. de Beaufort, and D. Fanelli. Doing the right thing: A qualitative investigation of retractions due to unintentional error. *Science and engineering ethics*, 24(1):189–206, 2018.

[4] S. Jo, I. Trummer, W. Yu, X. Wang, C. Yu, D. Liu, and N. Mehta. Verifying text summaries of relational data sets. In *SIGMOD*, pages 299–316, 2019.

[5] G. Karagiannis, M. Saeed, P. Papotti, and I. Trummer. Scrutinizer: A mixed-initiative approach to large-scale, data-driven claim verification. *PVLDB*, 13(12), 2020.

[3] `https://iea.org/reports/world-energy-outlook-2018`