SHORT-PAPER

Demonstrating CEDAR: A System for Cost-Efficient Data-Driven Claim Verification

THARUSHI JAYASEKARA, Cornell University, Ithaca, NY, United States

IMMANUEL TRUMMER, Cornell University, Ithaca, NY, United States

Published: 22 June 2025

Citation in BibTeX format

SIGMOD/PODS '25: International Conference on Management of Data
June 22 - 27, 2025
Berlin, Germany

Conference Sponsors:
SIGMOD

Demonstrating CEDAR: A System for Cost-Efficient Data-Driven Claim Verification

Tharushi Jayasekara
tkj27@cornell.edu
Cornell University
Ithaca, NY, USA

Immanuel Trummer
itrummer@cornell.edu
Cornell University
Ithaca, NY, USA

Abstract

We present CEDAR, a system for cost-efficient, data-driven claim verification using large language models (LLMs). Given a collection of text documents containing claims that can be verified from relational data, CEDAR maps claims to SQL queries for automated verification. The system employs multiple verification approaches, from cost-effective zero-shot LLM invocations to more accurate but expensive iterative, agent-based methods. CEDAR dynamically selects and combines these methods, optimizing for accuracy and cost through cost-based optimization and parameter tuning. Visitors can interact with CEDAR to observe its behavior across various scenarios and data sets. They can explore how the system prioritizes verification approaches, starting with inexpensive methods and escalating to more advanced ones as needed. Users can experiment with CEDAR by modifying claims, data sets, and tuning parameters, observing how these changes influence verification accuracy and cost. Additionally, visitors can upload their own data sets to experience CEDAR's capabilities in real-world claim verification tasks. The demonstration highlights CEDAR's adaptability and its ability to balance cost and accuracy in customizable claim verification workflows.

CCS Concepts

• **Information systems** → **Data management systems**; • **Human-centered computing** → **Natural language interfaces**.

Keywords

Fact Checking, Natural Language Processing, Large Language Models, Cost Optimization

ACM Reference Format:
Tharushi Jayasekara and Immanuel Trummer. 2025. Demonstrating CEDAR: A System for Cost-Efficient Data-Driven Claim Verification. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3722212.3725098

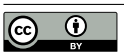
1 Introduction

Improperly analyzing data from large relational databases can produce incorrect textual summaries with repercussions across many industries such as scientific paper retractions in academia, financial losses in retail, flawed regulatory measures in policy-making as well as environmental damage in energy production. We propose CEDAR to address this problem of verifying claims on associated data sets. The input to the system is a collection of text documents and their associated data sets. The system identifies claims as correct or incorrect by translating the claims into SQL queries, executing them on the associated data set and comparing the query result to the value originally claimed on the claim.

Example 1.1. Consider the claim "The two fatal accidents involving Malaysia Airlines this year were the first for the carrier since 1995." extracted from a newspaper article on 538 [3]. This claim is derived from a data set, available on the 538 Web site. Our goal is to verify the claim value 'two' by generating an SQL statement that queries for the number of fatal accidents involving Malaysia Airlines on the aforementioned data, stored as table `airlines`. This claim maps to the SQL query `SELECT "fatal_accidents_00_14" FROM airlines WHERE airline = 'Malaysia Airlines'`. By executing this query and comparing its result with the claim value, we can verify whether the claim is *correct* or *incorrect*.

CEDAR utilizes agents powered by advanced LLMs, such as GPT-4, to translate claims into SQL queries. These agents, which use an iterative problem-solving approach, break complex tasks into smaller sub-tasks and leverage functions, or "tools," to solve them. For data-driven claim verification, the tools provide the agent with access to the relevant data and allow for comparisons between candidate SQL queries and claim values. It is crucial to prevent situations where the agent can exploit shortcuts to appear as though it has solved a task without actually producing a meaningful solution. For example, granting the agent access to the claim value might lead to SQL queries that simply return this claim value rather than capturing the intended semantics of the input claim. To address this, CEDAR carefully conceals critical information from the agent while ensuring sufficient context is provided to enable accurate translation of the claim into an SQL query. In a final post-processing step, CEDAR combines traces of tool usage from multiple iterations to construct a single SQL query corresponding to the input claim.

However, the iterative nature of agents increases the amount of text processed by the LLM, leading to higher verification costs since LLM fees are tied to the volume of input and output. To address this, CEDAR adopts a multi-stage verification strategy. Instead of immediately deploying the most resource-intensive methods like agents, it begins with less expensive, simpler approaches that are often sufficient. These include single-shot claim-to-SQL translation methods, which require only a single LLM invocation. To improve the success rate of these methods, CEDAR automatically gathers examples of accurately translated claims and uses them as prompts for few-shot learning.



SIGMOD-Companion '25, Berlin, Germany

ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3725098

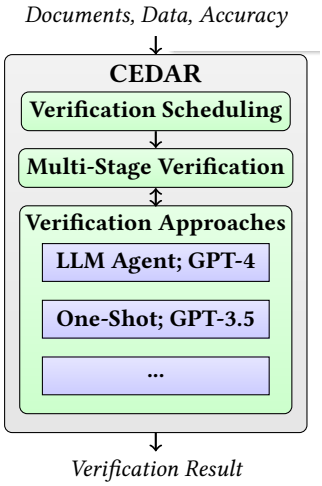


Figure 1: Architecture of CEDAR

Each of the described verification methods can be paired with various language models, resulting in a wide range of options that balance success probability—i.e., the likelihood of accurately translating a claim into an SQL query—and monetary costs. Since LLM outputs are inherently non-deterministic, producing different results for the same input across multiple attempts, it can be beneficial to retry the same method on a given claim. CEDAR leverages profiling data collected by testing different verification approaches on sample input data. Based on user-defined preferences for cost and quality tradeoffs, CEDAR identifies an optimized verification schedule that specifies which methods to apply and the number of retries to attempt before transitioning to the next approach.

To determine this schedule, CEDAR employs cost-based optimization. This problem is conceptually similar to optimizing the order of expensive predicates, accounting for both their costs and selectivity, but with the added complexity of allowing retries with potentially varying outcomes. CEDAR demonstrates that its cost function adheres to the principle of optimality and introduces a dynamic programming algorithm to compute the optimal schedule based on this cost model.

2 System Overview

Figure 1 shows an overview of the CEDAR system. Input to CEDAR is a set of text documents containing claims to be verified, along with their associated data sets. Additionally, users can tune a parameter to specify the accuracy threshold. Setting a higher value will make the verification results more accurate, but it will also cost more as it involves the use of more expensive verification methods. For each claim, CEDAR outputs the verification result (whether the claim is correct or incorrect), along with the SQL query used for verification.

Verification Approaches. CEDAR employs a range of verification techniques that balance cost and quality, including agent-based iterative methods and more affordable one-shot methods. All approaches utilize large language models for translating claims into SQL queries, with the current implementation leveraging OpenAI's GPT model series. The most basic and cost-effective method supported by CEDAR is one-shot translation, where the LLM is invoked a single time to convert a claim into an equivalent SQL query. This method can work with any LLM, and the current implementation uses both GPT-4 and GPT-3.5. However, this approach struggles with complex claims because the LLM often lacks sufficient information about the associated datasets (e.g., the exact names of constants in the data) to produce an accurate translation. Determining the required information for a given claim in advance is challenging, which motivates the use of an iterative approach.

The iterative method allows the LLM to make multiple attempts and request additional data when necessary. CEDAR employs the ReAct [22] framework for this purpose, where the LLM alternates between "reasoning" and "action." Through reasoning, the LLM can break down complex tasks into smaller sub-problems, and through action, it selects and invokes specific tools with task-relevant input parameters. Instead of producing a single query, the ReAct agent generates multiple SQL queries during the process. These queries often include intermediate results, such as constants derived from earlier steps, which are crucial for claim verification. In a final post-processing stage, CEDAR reconstructs a complete SQL query by combining fragments from all iterations of the ReAct agent.

Multi-Stage Claim Verification. The various verification approaches are applied with a specified number of retries until claims are accurately translated into SQL queries for verification. Previous research [11] indicates that incorrect claims often feature claim values that are close to the true values. This observation has also been utilized in earlier work on data-driven fact-checking [9]. Leveraging this insight, CEDAR evaluates the plausibility of the SQL queries generated during claim translation by checking whether their results are sufficiently close to the claimed values. If one approach fails to generate a plausible SQL query, CEDAR moves to the next verification approach based on a pre-determined schedule.

Verification Scheduling. In order to conduct multi-stage claim verification, CEDAR first determines an optimal order in which verification approaches are applied and the number of retries alloted to each. This sequence is critical as it significantly influences the overall cost of claim verification. To optimize this process, CEDAR employs cost-based optimization, utilizing profiling statistics to estimate the success probabilities and average costs of various methods. In addition, it incorporates user-defined preferences, such as the accuracy threshold, to balance the trade-off between cost and quality. The optimization algorithm eliminates schedules that are not Pareto-optimal based on cost and accuracy metrics. During a final selection phase, the algorithm focuses on schedules that meet the user-specified accuracy requirement, prioritize the use of the maximum number of verification methods, and minimize estimated costs.

CEDAR's approach uses dynamic programming, inspired by Selinger's classical algorithm for join ordering [18], but it introduces key differences. Unlike join ordering, CEDAR's search space includes not only the permutation of methods but also the number of retries for each approach, including the option to exclude certain methods entirely. Furthermore, the presence of multiple metrics such as cost and accuracy requires specialized pruning techniques and a tailored approach for the final schedule selection.

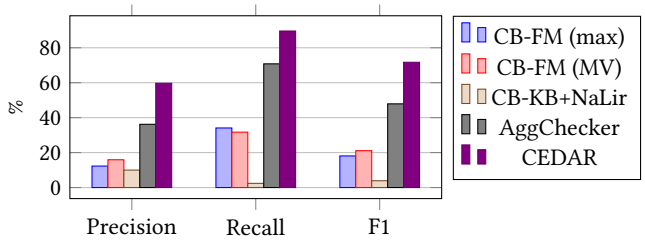


Figure 2: Comparing result quality of CEDAR and baselines.

3 Extract of Experimental Evaluation

Due to space limitations, we present a small extract of our experimental evaluation. All experiments are executed on a server with Intel Core i7-1355U CPU with 16 GB of main memory. CEDAR is implemented in Python, using the LangChain framework and OpenAI's GPT model series. CEDAR uses DuckDB to execute SQL queries. Unless noted otherwise, we use CEDAR with a high accuracy threshold of 99%.

We evaluate all baselines on a diverse collection of 56 data summaries, containing 392 claims in total. This data set was introduced in prior work [9] and covers publicly available articles from several newspapers (including 538 and the New York Times), as well as summaries of Stack Overflow developer surveys and Wikipedia articles.

We compare CEDAR to multiple prior systems for data-driven fact-checking. These include AggChecker [9], as well as the ClaimBuster system [5, 7]. While AggChecker focuses on data-driven fact-checking as well, ClaimBuster supports a broader range of claims. Following prior work on data-driven fact-checking [9], we compare to ClaimBuster in three different configurations.

Figure 2 compares output quality for all baselines. CEDAR outperforms the baselines for any of the three metrics: recall (the ratio of incorrect claims identified), precision (the ratio of claims marked as incorrect that are indeed incorrect), and the F1 Score (combining precision and recall). For instance, CEDAR improves F1 Scores from 48% (for the runner-up baseline in that category) to 72%. Comparing accuracy for baselines that translate claims to SQL queries (a subset of the baselines we consider), CEDAR infers correct SQL queries for 81% of claims (compared to only 58% for AggChecker). Clearly, integrating state-of-the-art LLM-based text analysis leads to impressive gains in verification quality.

4 Demonstration Setup

A live demo of CEDAR will be available to visitors to interact with, over a web interface running on a laptop connected to a portable projector. The laptop must have access to the Internet as it has to query the OpenAI API during the claim verification process. After a brief introduction to the system, visitors can proceed to interact with CEDAR themselves.

Figure 3 provides a comprehensive overview of the key features of the CEDAR system. The system accepts a document containing claims and its associated dataset as input. Users can either upload their own documents and datasets or select from a collection of 56 predefined scenarios sourced from platforms such as Wikipedia,

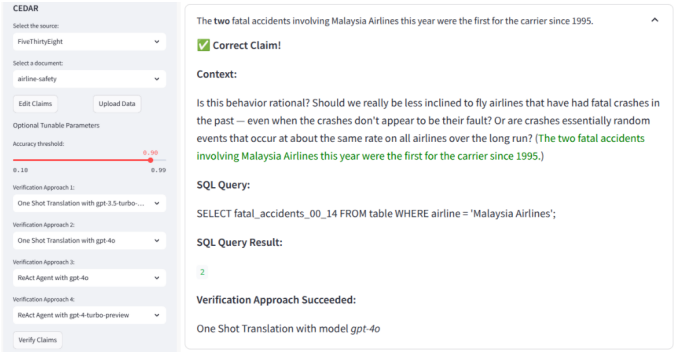


Figure 3: Interface for claim verification: users select a document or upload their own data, optionally configuring the tunable parameters to verify the claims in the document.

FiveThirtyEight, and StackOverflow. Each document in the system includes multiple claims that have been marked for verification. Once a document is selected, CEDAR initiates the verification process in the background. Within seconds, the claims are displayed sequentially on the interface as the system completes their verification. Claims are presented within expandable sections, with the claim value emphasized in bold. By clicking on a claim, users can expand it to access detailed verification results.

The verification output first indicates whether a claim is correct or incorrect, accompanied by the claim's context. The context includes the surrounding paragraph from the document, with the claim itself highlighted in green (for correct claims) or red (for incorrect claims). Additionally, the results display the SQL query used for verification and its corresponding output. Among the multiple verification methods available, the system identifies and highlights the approach that successfully verifies the claim. For verification approaches involving ReAct agents, the system provides an explanation of the agent's reasoning process. This feature allows users to observe how the agent decomposes complex tasks into smaller sub-problems and solves them step-by-step.

Finally, the system presents various usage statistics to enable a comparative analysis of different verification methods. These statistics include metrics such as the monetary cost per claim, time spent per claim, the frequency of successful verification approaches, total cost per approach, and total time spent per approach.

This is the general use case for interacting with CEDAR. Additionally, there are some customizations that enhance the utility of the demonstration. The interface includes a sidebar on the left featuring two key options: 'Edit Claims' and 'Upload Data'. The 'Edit Claims' option allows users to modify claims or dataset entries before initiating the verification process. Upon selecting this option, an editable table corresponding to the dataset of the selected document is displayed. Users can adjust claim values and rerun the verification process. This feature provides insights into how modifications to claims, such as altering a correct claim value to an incorrect one, can influence the verification process. For instance, incorrect claim values may trigger the use of more resource-intensive verification methods as CEDAR exhaustively attempts to validate the claim. The 'Upload Data' option enables users to upload custom

datasets in CSV format for verification. By utilizing this functionality, users can define their own claims and leverage CEDAR's verification capabilities to evaluate them. This customization allows users to explore CEDAR's adaptability to diverse scenarios.

Users are also provided with the ability to tune a set of parameters in CEDAR to optimize its performance according to specific requirements. One such parameter is the accuracy threshold, which allows users to manage the trade-off between cost and quality. For scenarios where frequent verifications are required and a small margin of error is acceptable, a lower accuracy threshold can be set. Based on the selected threshold, CEDAR dynamically adjusts the verification schedule to align with the specified accuracy constraints.

Additionally, users have the option to manually modify the verification schedule during the demonstration. This functionality allows users to experiment with custom schedules and compare their impact on various performance metrics, helping them to identify verification schedules that achieve an optimal balance between cost and performance.

5 Related Work

We tackle the challenge of data-driven fact-checking using relational data and compare our approach with several existing methods [5, 9] through experiments. Other related works address similar problems but make different assumptions. For example, Scrutinizer [10] assumes that domain-specific classifiers for claim-to-SQL translation are trained by fact-checkers. Similarly, TabFact [2] and TFV [1] presuppose the availability of labeled training data for translation. In contrast, CEDAR does not rely on any user-provided training data. Instead, it operates on zero-shot approaches, leveraging the capabilities of advanced LLMs, or uses one-shot claim-to-SQL translation, utilizing automatically generated samples.

Our approach complements previous methods that focus on related issues, such as identifying check-worthy claims [5, 6], analyzing the robustness of data-driven claims through query perturbations [20, 21], or applying automated fact-checking to non-relational data sources, such as knowledge bases [8, 19] or unstructured data [4, 13, 15]. Since CEDAR is focused on fully automated fact-checking, it complements methods aimed at assisting human fact-checkers [14]. Our work also relates to text-to-SQL translation techniques [12, 16, 17]. However, rather than translating isolated questions, we focus on translating claims embedded within larger text documents that include a claimed result. This claimed result plays a crucial role in our multi-stage approach, serving as a key signal for assessing the accuracy of query translation. Without leveraging the claimed result, multi-stage verification and scheduling, which are core components of our technical contribution, would not be feasible. By exploiting these claimed values, CEDAR ensures a more reliable verification process, strengthening confidence in the correctness of earlier translation stages.

6 Conclusion

The proposed demonstration focuses on CEDAR, a system for cost-efficient data-driven claim verification. Visitors will be able to interact with CEDAR under different scenarios, data sets and observe their influence on the performance metrics.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Award No. 2239326.

References

[1] Mingke Chai, Zihui Gu, Xiaoman Zhao, and Ju Fan. 2021. TFV : A Framework for Table-Based Fact Verification. *Data Engineering Bulletin* 59 (2021), 39–51.
[2] Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. TabFact: A Large-scale Dataset for Table-based Fact Verification. *CoRR* abs/1909.0 (2019), 1–17. arXiv:1909.02164 http://arxiv.org/abs/1909.02164
[3] FiveThirtyEight. 2014. Should Travelers Avoid Flying Airlines That Have Had Crashes in the Past? https://fivethirtyeight.com/features/should-travelers-avoid-flying-airlines-that-have-had-crashes-in-the-past/
[4] Jiahui Geng, Yova Kementchedjhieva, Preslav Nakov, and Iryna Gurevych. 2024. Multimodal Large Language Models to Support Real-World Fact-Checking. arXiv:2403.03627 [cs.CL] https://arxiv.org/abs/2403.03627
[5] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward automated fact-checking: detecting check-worthy factual claims by ClaimBuster. In *SIGKDD*. 1803–1812.
[6] Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015. Detecting check-worthy factual claims in presidential debates. In *CIKM*. 1835–1838. doi:10.1145/2806416.2806652
[7] Naeemul Hassan, Anil Nayak, Vikas Sable, Chengkai Li, Mark Tremayne, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, and Aaditya Kulkarni. 2017. ClaimBuster: the first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment* 10 (08 2017), 1945–1948. doi:10.14778/3137765.3137815
[8] Viet-Phi Huynh and Paolo Papotti. 2019. Buckle: evaluating fact checking algorithms built on knowledge bases. *VLDB* 12, 12 (2019), 1798–1801.
[9] Saehan Jo, Immanuel Trummer, Weicheng Yu, Xuezhi Wang, Cong Yu, Daniel Liu, and Niyati Mehta. 2019. Verifying Text Summaries of Relational Data Sets. In *SIGMOD '19*. 299–316. doi:10.1145/3299869.3300074
[10] Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. Scrutinizer: a mixed-initiative approach to large-scale, data-driven claim verification. *Proc. VLDB Endow.* 13, 12 (2020), 2508–2521. doi:10.14778/3407790.3407841
[11] Georgios Karagiannis, Immanuel Trummer, Saehan Jo, Shubham Khandelwal, Xuezhi Wang, and Cong Yu. 2020. Mining an "Anti-Knowledge Base" from Wikipedia Updates with Applications to Fact Checking and Beyond. *PVLDB* 13, 4 (2020), 561–573. https://doi.org/10.14778/3372716.3372727
[12] Fei Li and HV Jagadish. 2014. NaLIR: an interactive natural language interface for querying relational databases. In *SIGMOD*. 709–712.
[13] Miaoran Li, Baolin Peng, Michel Galley, Jianfeng Gao, and Zhu Zhang. 2024. Self-Checker: Plug-and-Play Modules for Fact-Checking with Large Language Models. In *Findings of the Association for Computational Linguistics: NAACL 2024 - Findings*. 163–181. doi:10.18653/v1/2024.findings-naacl.12
[14] Thanh Tam Nguyen, Matthias Weidlich, Hongzhi Yin, Bolong Zheng, Quoc Viet Hung Nguyen, and Bela Stantic. 2018. User guidance for efficient fact checking. *Proceedings of the VLDB Endowment* 12, 8 (2018), 850–863. doi:10.14778/3324301.3324303
[15] Dorian Quelle and Alexandre Bovet. 2024. The perils and promises of fact-checking with large language models. *Frontiers in Artificial Intelligence* 7 (2024). doi:10.3389/frai.2024.1341697
[16] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R Mittal, and Fatma Ozcan. 2016. ATHENA: An ontology-driven system for natural language querying over relational data stores. *VLDB* 9, 12 (2016), 1209–1220.
[17] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *EMNLP*. 9895–9901. doi:10.18653/v1/2021.emnlp-main.779 arXiv:2109.05093
[18] PG G Selinger, MM M Astrahan, D D Chamberlin, R A Lorie, and T G Price. 1979. Access path selection in a relational database management system. In *SIGMOD*. 23–34. http://dl.acm.org/citation.cfm?id=582095.582099
[19] Baoxu Shi and Tim Weninger. 2016. Fact Checking in Heterogeneous Information Networks. In *WWW*. 101–102. doi:10.1145/2872518.2889354
[20] Brett Walenz and Jun Yang. 2016. Perturbation analysis of database queries. *VLDB* 9, 14 (2016), 1635–1646.
[21] You Wu, Pk Agarwal, C Li, J Yang, and C Yu. 2014. Toward computational fact-checking. *VLDB* 7, 7 (2014), 589–600. doi:10.14778/2732286.2732295
[22] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL]