# OpenFactCheck: A Unified Framework for Factuality Evaluation of LLMs

**Hasan Iqbal**[1]* **Yuxia Wang**[1]* **Minghan Wang**[2]
**Georgi Georgiev**[3] **Jiahui Geng**[1] **Iryna Gurevych**[1] **Preslav Nakov**[1]
[1]MBZUAI [2]Monash University [3]Sofia University
{hasan.iqbal, yuxia.wang, preslav.nakov}@mbzuai.ac.ae

## Abstract

The increased use of large language models (LLMs) across a variety of real-world applications calls for automatic tools to check the factual accuracy of their outputs, as LLMs often hallucinate. This is difficult as it requires assessing the factuality of free-form open-domain responses. While there has been a lot of research on this topic, different papers use different evaluation benchmarks and measures, which makes them hard to compare and hampers future progress. To mitigate these issues, we developed **OpenFactCheck**, a unified framework, with three modules: (*i*) RESPONSEEVAL, which allows users to easily customize an automatic fact-checking system and to assess the factuality of all claims in an input document using that system, (*ii*) LLMEVAL, which assesses the overall factuality of an LLM, and (*iii*) CHECKEREVAL, a module to evaluate automatic fact-checking systems. OpenFactCheck is open-sourced[1] and publicly released as a Python library[2] and also as a web service[3]. A video describing the system is available at https://youtu.be/-i9VKL0HleI.

## 1 Introduction

Large language models (LLMs) have demonstrated impressive capabilities in generating naturally-sounding answers over a broad range of human inquiries. However, GPT-4o (OpenAI, 2023) and other text generation models still produce content that deviates from real-world facts (Bang et al., 2023; Borji, 2023; Guiven, 2023). This degrades the performance of LLMs and undermines their reliability, which is a significant bottleneck for their deployment (Chuang et al., 2023; Geng et al., 2023), especially for high-stake applications, e.g., clinical, legal, and financial settings.

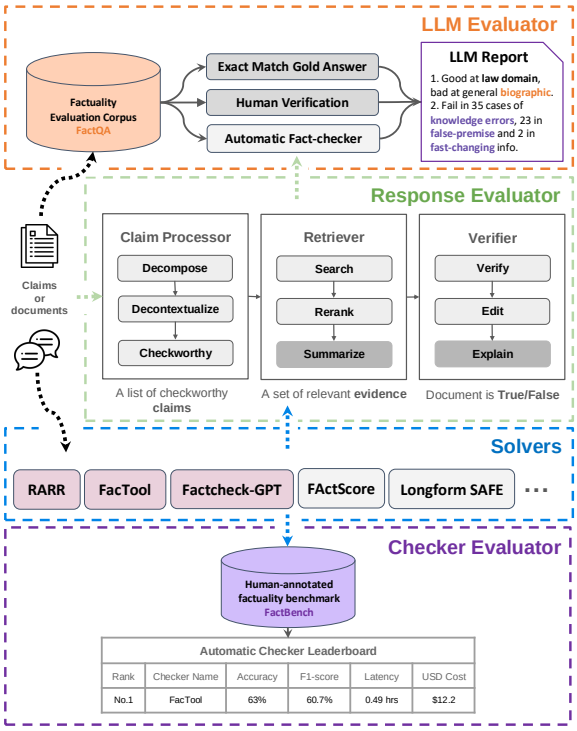


Figure 1: Overview of the OpenFactCheck demo system for LLM factuality evaluation and its modules. Green RESPONSEEVAL: a customized fact-checker to identify factual errors given text inputs. Orange LLMEVAL: an LLM factuality evaluator to assess the LLM factual ability from different aspects and then to produce a report to illustrate its weaknesses and strengths. Purple CHECKEREVAL: a fact-checker evaluator and leaderboard to encourage the development of advanced checkers in terms of performance, latency and costs.

Many studies have explored evaluating the factuality of LLMs (Lee et al., 2022; Chuang et al., 2023; Shi et al., 2023; Chen et al., 2023). Two challenges have been identified: (*i*) it is difficult to assess the factuality of open-domain free-form responses, and (*ii*) different papers use different evaluation datasets and measures, which makes it hard to compare them, thus hampering future progress (Wang et al., 2024c). To mitigate these issues, we introduce **OpenFactCheck**.

*Equal contribution.
[1]https://github.com/mbzuai-nlp/openfactcheck
[2]https://pypi.org/project/openfactcheck/
[3]http://app.openfactcheck.com



*Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*
*System Demonstrations*, pages 219–229
November 12-16, 2024 

OpenFactCheck is an Open-source Factuality Evaluation Framework for LLMs and it comprises the following three core modules (see Figure 1):

- RESPONSEEVAL: It allows users to customize an automatic fact-checker and to verify the factuality of all claims made in a free-form document to alleviate the first problem.
- LLMEVAL: A unified LLM factuality evaluation module which applies seven factuality-specific benchmarks to assess the LLM factuality ability from different aspects and then produces a report to illustrate the weakness and strength, tackling the second challenge.
- CHECKEREVAL: It assesses the verification accuracy of fact-checkers, equipped with a leaderboard in terms of accuracy, latency, and costs, aiming to encourage the development of advanced automatic fact-checking systems.

The modules are designed for seamless integration, each contributing to and enhancing the capabilities of the others. The results of human verification derived from LLMEVAL can be used as the benchmark for evaluating the accuracy of automated fact-checkers. Simultaneously, the most effective checker identified in CHECKEREVAL can be deployed for automated fact-checking tasks. Each fact-checker in CHECKEREVAL can be an implementation in RESPONSEEVAL. Complex user inquiries may be considered as potential candidates of the factuality assessment dataset utilized in LLMEVAL.

**General users** can tailor their checkers according to their specific needs, such as domain specialization, cost-effectiveness, or rapid processing, and identify factual errors for both human-written text (a claim or document) and the outputs of LLMs. **LLM researchers and practitioners** can directly submit their LLM responses to the LLMEVAL by downloading our question set. Subsequently, we conduct evaluations to assess the model's factual accuracy and to generate a report analyzing the model performance from multiple aspects. Similarly, **developers** who seek to evaluate and to fairly compare the efficacy of their fact-checking systems to other ones can upload their checker's verification outcomes to CHECKEREVAL. Then, our system will show the ranking information in the leaderboard after evaluating under the same measurements.

To sum, three modules of OpenFactCheck respectively address the following:

- how to effectively identify factual errors in a text input;
- how to systematically evaluate the factuality ability of an LLM;
- which automatic fact-checker is the best, and which component dominates the final verification accuracy.

We have launched an open-source initiative that includes the development of a Python library and a web interface tailored to support three major functionalities. This foundation is expected to act as a catalyst for future advancements in factuality evaluation for LLMs. We encourage extensive implementation of unique, effective, and robust claim processors, retrievers and verifiers within fact-checking pipelines, collections of challenging questions that LLMs tend to make factual errors, and human-annotated fine-grained verification examples. We believe that this will help to promote and to advance future research on LLM factuality.

## 2 Related Work

While numerous automatic fact-checking systems have developed, such as *RARR*, *FactScore*, *FacTool*, *Factcheck-GPT*, *Longform SAFE* and *FIRE* (Gao et al., 2022; Min et al., 2023; Chern et al., 2023; Wang et al., 2023; Wei et al., 2024; Xie et al., 2024), they are often inaccessible to general users who lack a Python environment to compile code and run verification. Although these systems can function as the backend of a service, a user-friendly web interface is necessary to allow general users to verify text inputs by simply typing or copying text and clicking a check button. OpenFactCheck addresses this by providing an accessible web interface.

In addition, various fact-checking systems have distinct advantages. For instance, *Factcheck-GPT* offers a fine-grained framework to involve all possible subtasks that could improve the fact-checking system, *FacTool* uses a low-latency evidence retriever through asynchronous processing, and *FactScore* introduces a scoring metric that calculates the percentage of true claims in a given text, thereby quantitatively assessing the credibility of the input. OpenFactCheck integrates these components into a unified system (Wang et al., 2024c).

Recent open-sourced demo system Loki (Wang et al., 2024a) also aims to leverage strength of various automatic fact-checkers, while it emphasizes optimization a single fact-checking system in terms of accuracy, latency, robustness, cost-efficiency, and extensive support for multiple languages and LLMs. In contrast, OpenFactCheck is a unified framework to cover three major functionalities for factuality evalaution of LLMs, including customizing a fact-checker by combining modules of different checkers, assessing LLM factuality from various perspectives, and evaluating the accuracy of automatic fact-checkers (Wang et al., 2024b).

## 3 System Architecture

The design of OpenFactCheck emphasizes two principles: *(i)* customizability and extensibility for both users and developers, and *(ii)* compatibility with existing methods and datasets. It consists of three modules: RESPONSEEVAL, LLMEVAL, and CHECKEREVAL. We detail the design and implementation of each components below.

### 3.1 RESPONSEEVAL

RESPONSEEVAL allows users to build a customized fact-checking system by selecting a claim processor, a retriever, and a verifier in web pages. The current version supports the following fact-checking systems: *RARR*, *FacTool* and *Factcheck-GPT* (Gao et al., 2022; Chern et al., 2023; Wang et al., 2023).

**Configurable Architecture** We consolidate various fact-checking systems into a three-step process, encapsulated by three classes: `claim_processor`, `retriever`, and `verifier` (Wang et al., 2024c). These classes are instantiated and sequentially connected to form a pipeline that addresses the following tasks: *(i)* breaking down a document into individual claims, *(ii)* gathering pertinent evidence for each claim, and *(iii)* evaluating the veracity of each claim based on the evidence provided. This sequence of tasks is referred to as `solvers`. (see the pseudo code in Appendix A)

The implementation of a task solver can be flexible, just ensuring that the input and the output are aligned with the abstract class definitions. For example, evidence can be retrieved by calling SerpAPI or by searching Wikipedia using BM25, but we must return a list of relevant passages given an input claim.

Moreover, task solvers in our pipeline are not hard-coded, but can be configured through a *YAML* configuration file. Thus, users can combine task-solver implementations from different systems (e.g., using *Factcheck-GPT*'s claim processor, *RARR*'s retriever, and *FacTool*'s verifier) and start the verification from any step. For example, users can start from the step of retrieval when the input does not need decomposition.

This functionality is achieved by a message-passing mechanism, where a `success_flag` is used to indicate whether the current task solver successfully executes and returns the expected output. The success flag passes through the pipeline as the configured order of solvers, guaranteeing that the output of the preceding solver fits the input for the current solver, otherwise error warning will be issued. Practically, the input and the output parameter names for the task solvers are defined in the configuration file. To link different solvers into a pipeline, one only needs to ensure that the current solver output name matches the input name of the succeeding solver. A `FactcheckerState` class ensures storage of all information in the verification.

**Extendable Architecture** Inspired by Fairseq, our framework is designed to be highly extendable by treating any third-party task solvers as plug-ins (Ott et al., 2019). As long as the developed task solvers adhere to our class interface definitions, they can be imported and used in our framework.

### 3.2 LLMEVAL

We observed that studies assessing language models' factuality or evaluating whether the methods are effective to mitigate model hallucinations use different datasets and metrics. This makes it difficult to compare, in the same conditions, the factuality of different models as well as to compare the effectiveness of different factuality enhancement approaches. Moreover, a lot of prior work applied datasets such as MMLU (Hendrycks et al., 2021), StrategyQA (Geva et al., 2021) and HotpotQA (Yang et al., 2018) to evaluate model's factuality. These datasets tend to focus on assessing the general performance, rather than factuality. To this end, we first collect a dataset *FactQA* by gathering factual questions of existing datasets that are curated to probe diverse factual errors and span across a spectrum of domains, to fairly evaluate LLMs' factuality under the same criteria

| Dataset↓ | The Ability to Evaluate | Domain | Error | Size |
|---|---|---|---|---|
| **Snowball** | Snowballing hallucination when model immediately output | Math, history, graph search | Type 2 | 1,500 |
| **SelfAware** | Understand their own limitations on the unknowns | Biology, philosophy, psychology, history | Type 1,3 | 3,369 |
| **FreshQA** | Answer questions changing fast over time or with false premises | Sports, entertainment, history, technology | Type 3 | 600 |
| **FacTool-QA** | Respond knowledge-based questions | History, geography, biology, science | Type 1 | 50 |
| **FELM-WK** | Answer world-knowledge questions | History, biology, geography, sports | Type 1 | 184 |
| **Factcheck-Bench** | Answer open-domain, false-premise questions | Technology, history, science, sports | Type 1,2 | 94 |
| **FactScore-Bio** | Generate detailed biographies | Biography | Type 1,3 | 683 |
| **Total** | LLM factuality against world knowledge | 482 domains, top20 accounts for 70% | Type 1,2,3 | 6,480 |

Table 1: **FactQA**: factual vulnerability, domain, potential error type and size across seven component datasets.

**Factual Question Collection** We collected factual questions from seven commonly-used corpora that is collected deliberately to assess LLM's factuality, including Snowball (Zhang et al., 2023a), SelfAware (Yin et al., 2023), FreshQA (Vu et al., 2023), *FacTool* (Chern et al., 2023), FELM-WK (Chen et al., 2023), *Factcheck-GPT* (Wang et al., 2023) and FactScore-Bio, a total of 6,480 examples shown in Table 1, referring to FactQA (see dataset details in Appendix C).

To concretely analyze models' vulnerability, we identify three labels for each question from the perspective of the knowledge domain, the topic, and the potential error type if a LLM generates a factually incorrect response. So each example includes the following fields: *question*, *domain*, *topic*, *ability to test*, *task* and *source*. Domains involve general, legal, biomedical, clinical, scientific and so on. Given a domain, we further fine-grained topics. Three common error types are presented.

***Type1*: *Knowledge error*** is the most common error when the model produces hallucinated or inaccurate information due to lacking relevant knowledge or internalizing false knowledge in the pre-training stage or in the alignment process.

***Type2*: *Over-commitment error*** occurs when the model fails to recognize the falsehoods (or jokes) in the prompt or previously-generated context, and provides an inaccurate or inappropriate response.

***Type3*: *Disability error*** happens when the model is unable to search up-to-date information to correctly answer questions whose answers change over time, e.g., *What is today's gas price in New York* (fast-changing). See more in Appendix B.

**Evaluation Measurement** For questions that can be answered by Yes/No or have a short gold answer, we perform exact matching between the model responses and the gold standard answer to judge whether the response is factually correct, and then to calculate accuracy, such as for Snowball and SelfAware.

| Dataset ↓ | #True | #False | #Unknown | Total |
|---|---|---|---|---|
| FacTool-QA | 177 | 56 | 0 | 233 |
| FELM-WK | 385 | 147 | 0 | 532 |
| Factcheck-Bench | 472 | 159 | 47 | 678 |
| HaluEval | 3,692 | 815 | 0 | 4,507 |

Table 2: The number of true, false claims and unknown (no-enough-evidence or opinions) for FacTool-QA, FELM-WK and Factcheck-Bench, the number of responses for HaluEval (no claim-level labels).

For FreshQA, we use the *FreshEval* proposed in Vu et al. (2023) to evaluate the correctness of model's responses. For open-domain questions from the other four datasets with free-form and long responses, there are no gold standard answers. We use automatic fact-checking systems to judge the correctness of claims and obtain the percentage of true claims as the accuracy for a response.

### 3.3 CHECKEREVAL

Automatic fact-checking systems aim to identify whether a claim or a document is true or false, but the results are not necessarily correct. To assess the accuracy of automatic fact-checkers, we gather four LLM factuality benchmarks with human-annotated factual labels for three levels of granularity text: claims/segments/documents given (question, ChatGPT response) pairs, including FacTool-QA, FELM-WK, Factcheck-Bench and HaluEval as shown in Table 2. We refer to them as FactBench. We use precision, recall, and F1-score with respect to the *True* or *False* claim/document to evaluate the effectiveness of fact-checking systems.

**Discussion about *Unifying*** It can be argued that the underlying philosophies of the three modules differ, reflecting varying interpretations of factuality. For example, the design view o LLMEVAL and CHECKEREVAL differs from that of RESPONSEEVAL.

Our goal is to integrate all LLM factuality evaluation functionality into a unified framework, while preserving the individual function.

The LLMEVAL employs different metrics across datasets. This may be debated. Similarly, we aim to consolidate these datasets into a unified benchmark, enabling other studies to utilize a standardized evaluation function. This approach would enhance the fairness of comparisons across studies, as they would be evaluated consistently, despite the use of dataset-specific evaluation measures.

We acknowledge the limitation of the current CHECKEREVAL, which is restricted to evaluating the verification step. We plan to progressively extend its capabilities to support fine-grained evaluations across multiple steps.[4]

## 4 Access and Deployment

OpenFactCheck is accessible via a user-friendly web interface and features an integrated database that maintains a user leaderboard. It is also available as a standalone open-source Python library.

### 4.1 Python Library

OpenFactCheck is available as an open-source Python library on PyPI, designed for flexibility and ease of integration into existing projects. This library equips developers with essential components for fact-checking in any Python environment, making it an optimal choice for enhancing applications with fact-checking features. The library employs a fluent interface to ensure its usage is intuitive for both beginners and experts alike.

Users can install the library by simply using the pip package manager:

```
$ pip install openfactcheck
```

The library includes detailed documentation to assist developers in customizing and extending the functionality to meet their specific needs and it is continually updated to ensure compatibility with the latest research and data security standards.

**Usage Examples** The first step is to import the necessary library components and initialize `OpenFactCheckConfig` configuration and `OpenFactCheck` class, which requires no input values for default usage, as shown below:

```
from openfactcheck import OpenFactCheck,
↪  OpenFactCheckConfig
config = OpenFactCheckConfig()
ofc = OpenFactCheck(config)
```

Upon importing the library, users are required to secure API keys from platforms utilized by OpenFactCheck's default solvers for evidence retrieval and claim verification. These keys are available from OpenAI[5], SerpAPI[6], and ScraperAPI[7]. After acquiring the keys, they need to be configured as environment variables to enable their use within the library.

The three key functionalities outlined in Section 3 are implemented as shown in Figure 2. We can see that the design of the library is intuitive and straightforward, enabling users to apply it without extensive learning, and practioners to perform further developments easily (e.g., reusing one example by simply altering the evaluator name in each instance). The intermediate results are also logged on the terminal and are omitted here for brevity.

User is provided with the benchmarks for the LLM and FactChecker evaluations, and can upload the responses to the library for evaluation in the form of CSV files. CSV file format for LLM evaluation has two columns: `index` and `response`, while the FactChecker evaluation CSV file format has three columns: `label`, `time`, and `cost`.

### 4.2 Web Interface

The web interface of OpenFactCheck provides a user-friendly platform that allows general users to interactively engage with the fact-checking functionalities. It is designed to accommodate both novice and expert users, facilitating easy access to the comprehensive evaluations involved in the assessment of LLM factuality. The web interfaces are organized into four distinct sections as illustrated in Figure 3 (a).

In RESPONSEEVAL page as shown in Figure 3 (b), users can click the dropdown list to select from a range of pre-implemented `claim processor`, `retriever`, and `verifier`. Then, users can input text either written by human or generated by machine into the text box and click *Check Factuality* to obtain the verification results. As the example demonstrated in the Figure, it includes two claims.

[4]The evaluator currently supports both claim-level and document-level verification, depending on whether users download claim or document datasets.

[5]https://openai.com/api
[6]https://serpapi.com
[7]https://scraperapi.com

```
ofc.ResponseEvaluator.evaluate(response: str)
# response: string output from LLM
```

```
ofc.LLMEvaluator.evaluate(model_name: str,
↪  input_path: str)
# model_name: evaluated model name.
# input_path: path to the CSV containing
↪  responses for the LLM Benchmark.

# Output
# A dictionary with detailed scores (precision,
↪  recall, f1, accuracy, cost, time etc. for
↪  each dataset subset i.e. snowballing,
↪  selfaware, freshqa, factoolqa, felm-wk,
↪  factcheck-bench and factscore-bio.
```

```
ofc.CheckerEvaluator.evaluate(input_path: str)
# input_path: path to the CSV containing
↪  responses for the FactChecker Benchmark

# Output
# A dictionary with detailed scores (precision,
↪  recall, f1, accuracy, cost, time etc.)
```

Figure 2: Usage examples of three major modules: RESPONSEEVAL, LLMEVAL and CHECKEREVAL.

The system collected sixteen pieces of evidence, and one claims is supported and one claim is refuted, resulting the overall credibility of 50% and judgement "False" for this whole input.

For both the LLMEVAL and RESPONSEEVAL pages exhibited in Figure 3 (d), users first download either the question set FactQA or the claims/documents in FactBench. After being ready to upload the responses of the LLM that users aim to assess or the verification results of the fact-checkers to test, users type their details including name, email address and so on, and provide the option to opt in or out of leaderboard inclusion (see Figure 3 (d)). If users agree, their information and rank will be displayed on the leaderboard, otherwise invisible for others.

It may takes some time for LLMEVAL to generate teh evaluation report, depending on the system's current load. Once the report is ready, it is emailed directly to the user, eliminating the need to wait within the application. LLM factuality evaluation report presents LLM factuality from various aspects, and specifically includes accuracy and confusion matrix of short answers, pie chart indicating accuracy over fresh questions and bar chart showing the percentage of true, false, controversial claims for free-form responses, as shown on Figure 3 (e).

Similarly, CHECKEREVAL results present the number of evaluated examples, the overall accuracy, total time and USD cost, fine-grained precision, recall and F1-score for false and true classes, and a confusion matrix showing the misidentification of this fact-checker. The submission in Figure 3 (f) reveals that this checker performs equally poor over both false and true claims in verification. This evaluation is instant. [8]

## 5 Conclusion and Future Work

We implemented a unified, easy-to-use and extensible framework OpenFactCheck. It is accessible by both Python library and web service, supporting the customization and evaluation of automatic fact-checking systems and LLM factuality evaluation. Specifically, OpenFactCheck allows general users to check whether a claim and a document are factual or not by clicking **Check**, and also facilitate LLM practitioners and developers to effectively and efficiently evaluate the factuality of their LLMs from various perspectives, and to assess the accuracy of automatic fact-checking systems. In the future, we will continue to integrate new techniques, features, and evaluation benchmarks to OpenFactCheck to facilitate the research progress of LLM fact-checking.

## Limitations and Future Work

While OpenFactCheck presents a comprehensive framework for factuality evaluation of LLMs, several limitations must be acknowledged:

**Multilingual Expansion** OpenFactCheck is a platform that combines the features of various fact-checking systems and is designed to be language-agnostic. While the default task solvers in the system are configured for English, the platform can be expanded to accommodate other languages by developing task solvers that align with the specific linguistic requirements of those languages. This flexibility allows for easy adaptation and extension to support multilingual fact-checking capabilities.

**Evaluation Datasets** The effectiveness of OpenFactCheck is dependent on the quality and diversity of the datasets used for evaluation. While we have integrated multiple datasets to cover a broad spectrum of domains and potential factual errors, the evaluation is still limited by the inherent

[8] See more evaluation results in Wang et al. (2024b).

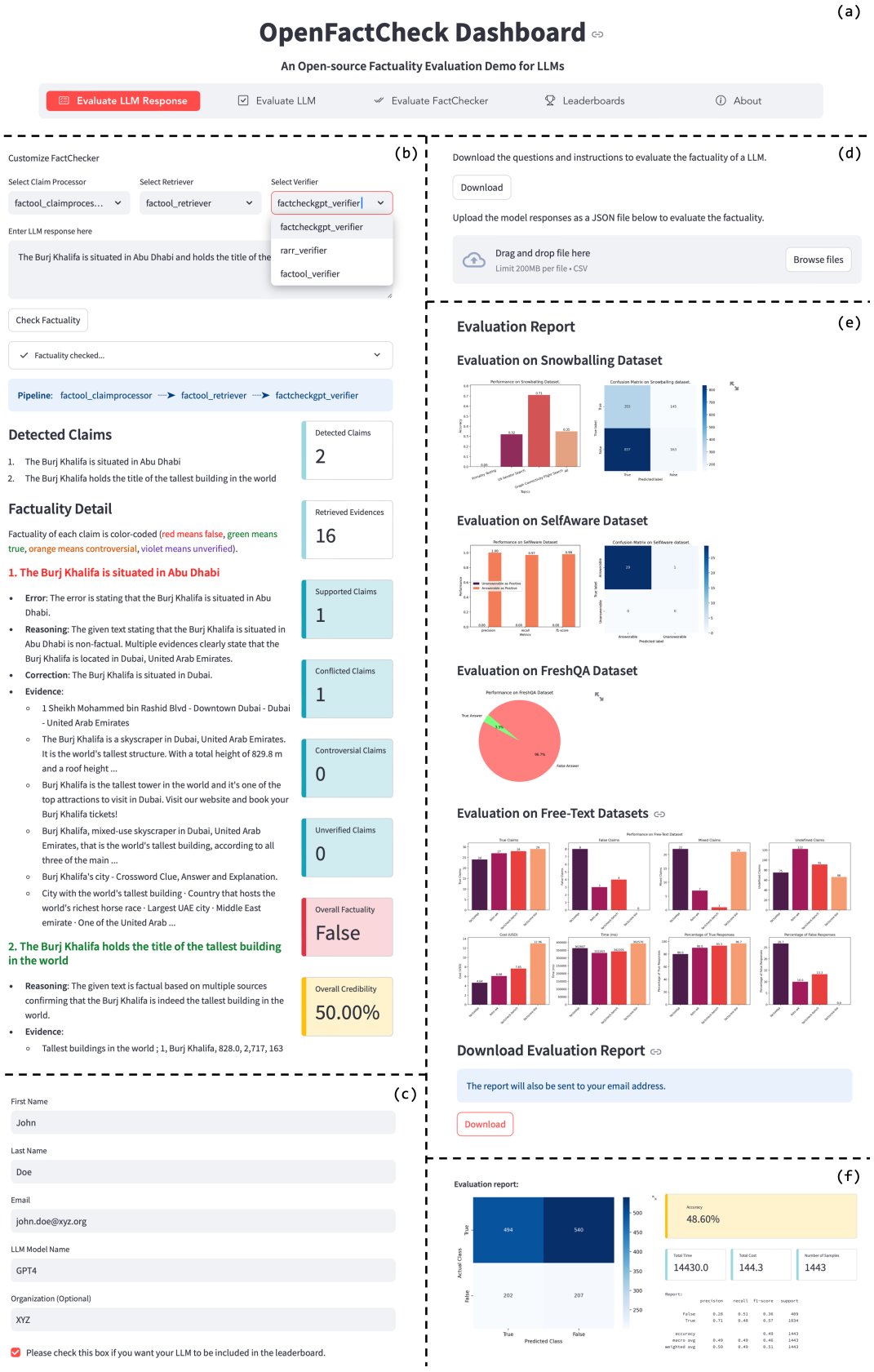


Figure 3: OpenFactCheck Dashboard: **(a)** is the navigation bar. **(b)** a claim processor breaking down the input into two atomic claims. The retriever collected 16 pieces of evidence, and the verifier assessed each claim individually, with one true and one false, resulting 50% credibility overall. **(c)** shows the user information required before uploading LLM responses or verification results to LLMEVAL and CHECKEREVAL. **(d)** shows the functions of downloading and uploading. **(e)** and **(f)** exhibit the LLM and FactChecker Evaluation report respectively.

biases and coverage gaps in these datasets. For instance, some specialized domains may not be adequately represented, potentially affecting the robustness of the evaluation for LLMs in those areas.

**Latency and Costs** The performance of automatic fact-checking systems integrated within OpenFactCheck can vary significantly in terms of latency and operational costs. High accuracy often comes at the expense of increased computational resources and processing time, which may not be feasible for all users, particularly those with limited budgets or time constraints.

**Reliance on External Knowledge Sources** The fact-checking modules depend heavily on external knowledge sources, such as Wikipedia and web search engines. The availability and reliability of these sources can affect the accuracy and completeness of the fact-checking process. Furthermore, the dynamic nature of web content means that the information retrieved may not always be up-to-date.

**Temporal Issues** The factuality of statements can change over time due to evolving events, new discoveries, or updated information. OpenFactCheck does not explicitly account for temporal dynamics as of now, which may lead to discrepancies between the evaluation results and the current state of knowledge. Authors are already working on factuality evaluation methods that consider temporal aspects, which will be integrated into OpenFactCheck in future releases.

## Ethical Statement

The development and deployment of OpenFactCheck are guided by a commitment to ethical principles, ensuring that the framework is used responsibly and for the benefit of society:

**Transparency and Accountability** We strive to maintain transparency in the design, implementation, and evaluation of OpenFactCheck. The source code and datasets are publicly available, enabling scrutiny and fostering trust within the research community. We encourage users to report any issues or biases they encounter, facilitating continuous improvement.

**Bias Mitigation** Recognizing that biases can exist in both datasets and LLMs, we are dedicated to minimizing such biases in OpenFactCheck. By integrating diverse evaluation benchmarks and encouraging the development of fair fact-checking approaches, we aim to reduce the impact of biases on factuality evaluation outcomes.

**Social Impact** By enhancing the factual accuracy of LLMs, OpenFactCheck aims to contribute positively to society. Accurate information is crucial for informed decision-making and public discourse. We believe that improving the reliability of LLM outputs can help combat misinformation and support the dissemination of truthful information.

## References


Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. *ArXiv preprint*, abs/2302.04023.

Ali Borji. 2023. A categorical archive of ChatGPT failures. *ArXiv preprint*, abs/2302.03494.

Shiqi Chen, Yiran Zhao, Jinghan Zhang, I-Chun Chern, Siyang Gao, Pengfei Liu, and Junxian He. 2023. FELM: Benchmarking factuality evaluation of large language models. *ArXiv preprint*, abs/2310.00741.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. 2023. FacTool: Factuality detection in generative AI - A tool augmented framework for multi-task and multi-domain scenarios. *ArXiv preprint*, abs/2307.13528.

Yung-Sung Chuang, Yujia Xie, and Hongyin Luo et al. 2023. DoLa: Decoding by contrasting layers improves factuality in large language models. *ArXiv preprint*, abs/2309.03883.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Y Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. 2022. Attributed text generation via post-hoc research and revision. *ArXiv preprint*, abs/2210.08726.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. 2023. A survey of language model confidence estimation and calibration. *ArXiv preprint*, abs/2311.08298.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Guiven. 2023. Llm failure archive (chatgpt and beyond). https://github.com/giuven95/chatgpt-failures.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Nayeon Lee, Wei Ping, and Peng et al. Xu. 2022. Factuality enhanced language models for open-ended text generation. *NeuralPS*, 35:34586–34599.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *ArXiv preprint*, abs/2305.14251.

OpenAI. 2023. GPT-4 technical report. *ArXiv preprint*, abs/2303.08774.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Weijia Shi, Xiaochuang Han, and et al. 2023. Trusting your evidence: Hallucinate less with context-aware decoding. *ArXiv preprint*, abs/2305.14739.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, et al. 2023. FreshLLMs: Refreshing large language models with search engine augmentation. *ArXiv preprint*, abs/2310.03214.

Hao Wang, Yuxia Wang, Minghan Wang, Yilin Geng, Zhen Zhao, Zenan Zhai, Preslav Nakov, Timothy Baldwin, Xudong Han, and Haonan Li. 2024a. Loki: An open-source tool for fact verification.

Yuxia Wang, Revanth Gangi Reddy, Zain Muhammad Mujahid, Arnav Arora, Aleksandr Rubashevskii, Jiahui Geng, Osama Mohammed Afzal, Liangming Pan, Nadav Borenstein, Aditya Pillai, et al. 2023. Factcheck-GPT: End-to-end fine-grained document-level fact-checking and correction of llm output. *ArXiv preprint*, abs/2311.09000.

Yuxia Wang, Minghan Wang, Hasan Iqbal, Georgi Georgiev, Jiahui Geng, and Preslav Nakov. 2024b. OpenFactCheck: A unified framework for factuality evaluation of llms. *ArXiv preprint*, abs/2405.05583.

Yuxia Wang, Minghan Wang, Muhammad Arslan Manzoor, Georgi Georgiev, Rocktim Jyoti Das, and Preslav Nakov. 2024c. Factuality of large language models in the year 2024. *ArXiv preprint*, abs/2402.02420.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. 2024. Long-form factuality in large language models. *ArXiv preprint*, abs/2403.18802.

Zhuohan Xie, Rui Xing, Yuxia Wang, Jiahui Geng, Hasan Iqbal, Dhruv Sahnan, Iryna Gurevych, and Preslav Nakov. 2024. FIRE: Fact-checking with iterative retrieval and verification. ArXiv preprint, under submission.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665, Toronto, Canada. Association for Computational Linguistics.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2023a. How language model hallucinations can snowball. *ArXiv preprint*, abs/2305.13534.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023b. Siren's song in the AI ocean: A survey on hallucination in large language models. *ArXiv preprint*, abs/2309.01219.

## A Pseudo Code of RESPONSEEVAL

In this section, we present the pseudo code for the RESPONSEEVAL, a modular system designed to process, retrieve, and verify claims found in textual documents. The system is divided into three primary components: the claim processor, the retriever, and the verifier. Each module is tasked with a specific function—extracting claims from the input document, retrieving relevant evidence, and verifying the factual accuracy of the claims, respectively. Figure 4 outlines the pseudo code implementation of each module, showcasing the flow from document processing to final verification. This structured approach allows for a systematic handling of claims, leveraging both natural language processing tools and deep learning models to ensure a comprehensive evaluation of document veracity.

## B Factual Error Evaluation

***Type1*: *Knowledge error*** is the most common error, occurring when the model produces hallucinated or inaccurate information. However, LLMs do not know what they do not know, sometimes overestimate their capacities and confidently output unknown information, leading to false responses. Mitigating such errors require: (a) learning and correcting parametric knowledge through the curation of corpora used in pre-training, supervised fine-tuning (SFT) and alignment, (b) augmenting by external knowledge in inference, (c) calibrating models to be aware of unknowns, and (d) configuring the decoding strategies (sample/beam-search, temperature), balancing diversity and accuracy (Zhang et al., 2023b).

***Type2*: *Over-commitment error*** occurs when the model fails to recognize the falsehoods (or jokes) inherent in the prompt or previously-generated context, and provides an inaccurate or inappropriate response. The left-to-right generation strategy used by LLMs poses potential risks that LLMs sometimes over-commit to the false premise in the context, even when they recognize they are incorrect (Zhang et al., 2023b). To address this issue, engineering better prompts is helpful, such as explicitly instructing models to first detect false premises in the prompt (Vu et al., 2023) and asking the same question in a different way (*Is 10733 a prime number? → What are the factors of 10733? Let's think step-by-step.*)

```
def claim_processor(document: str) ->
↪  List[str]:
    # FactScore
    paragraphs = documents.split("\n")
    sentences = [NLTK(para) for para in
    ↪  paragraphs]
    claims = [call_LLM(sentence,
    ↪  prompt="decompose into atomic claims")
    ↪  for sentence in sentences]

    # FacTool
    claims = call_LLM(document, promot="extract
    ↪  context-independent atomic claims based
    ↪  on the document")

    return claims

def retriever(claim: str, database: DB,
↪  retrieval_strategy: obj, search_api_key:
↪  str) -> List[str]:
    # offline DB dump
    evidence = retrieval_strategy(claim,
    ↪  database)

    # online web pages by calling API
    evidence = serper_or_serpapi(claim,
    ↪  search_api_key)

    return evidence

def verifier(claim: str, evidence: List[str])
↪  -> bool:
    # call LLMs
    factual_label = call_LLM(claim, evidence,
    ↪  prompt="based on the evidence and your
    ↪  own knowledge, determine whether the
    ↪  claim is true or false.")

    # use NLI models
    stance2factual = {
        "entailment": true,
        "contradiction": false,
        "neutral": "not enough evidence"
    }
    stances = [nli(evid, claim) for evid in
    ↪  evidence]
    majority_stance =
    ↪  majority_vote(factual_labels)
    factual_label =
    ↪  stance2factual[majority_stance]

    return factual_label
```

Figure 4: Pseudo code for classes in RESPONSEEVAL.

***Type3*: *Disability error*** happens when the model is unable to search up-to-date information to correctly answer questions whose answers change over time, e.g., *What is today's gas price in New York* (fast-changing). Retrieving external knowledge and augmenting it in the context would help for such cases. Note that we do not consider *reasoning errors* that arise when a claim is based on flawed reasoning or faulty logic.

| Domain | Size | Domain | Size |
|---|---|---|---|
| History | 771 | Science | 143 |
| Biography | 683 | Physics | 136 |
| Mathematics | 612 | Social Sciences | 111 |
| Transportation | 519 | Literature | 100 |
| Biology | 259 | Geography | 87 |
| Philosophy | 229 | Astronomy | 82 |
| Technology | 208 | Economics | 69 |
| Entertainment | 191 | Music | 66 |
| Psychology | 169 | Religion | 63 |
| Sports | 157 | General Knowledge | 53 |
| **Total** | | | **4,523 (69.8%)** |

Table 3: FactQA's top-20 domains and the number of examples from each domain.

Thus, ex exclude *irrelevant error* concerning that the content is unrelated to the question (Chen et al., 2023). The former highlights LLM's reasoning ability, which is more reflected in math and reasoning tasks, and the latter has more to do with response's helpfulness or human preference. They are important in LLM evaluation, and may implicitly influence factuality, but we will first focus on explicit causes, leaving the implicit for future work.

## C FactQA Component Datasets

**Snowball** dataset (Zhang et al., 2023a) comprises three question-answering subsets: primality testing, senator search, and graph connectivity, each with 500 yes/no questions. They aim to investigate snowballing hallucination when a model immediately outputs an incorrect answer (yes or no) as false generated context. Language models are prompted to first output a yes/no answer and then to provide explanations. When the immediate answer is wrong, the model tends to continue to snowball the false statements instead of correcting them.

**SelfAware** (Yin et al., 2023) aims to evaluate LLMs' ability to understand their own limitations and unknowns. This is achieved by assessing models' ability to identify unanswerable or unknowable questions. They compiled a collection of 1,032 unanswerable questions from online platforms like Quora and HowStuffWorks. In addition, they gathered 2,337 answerable questions from sources such as SQuAD, HotpotQA, and TriviaQA, resulting in a total of 3,369 questions.

**FreshQA** (Vu et al., 2023) is composed of 600 natural, open-ended questions, segmented into four primary categories based on the answer's stability: *never-changing*, for answers that rarely alter, *slow-changing*, for those that evolve over several years, *fast-changing*, for answers that shift within a year or less, and *false-premise*, encompassing questions with factually incorrect premises that need to be countered.

**FacTool** (Chern et al., 2023) detected factual errors in LLM generations across four different tasks: knowledge-based QA, code generation, mathematical reasoning, and scientific literature review. We used 50 knowledge-based QA FacTool-QA in FactQA.

**FELM** (Chen et al., 2023) collects responses generated from LLMs and annotated factuality labels in a fine-grained manner. The dataset consists of 5 categories, with examples per category as follows: 194 math, 208 reasoning, 125 science, 184 world knowledge (wk), and 136 writing recordings. We used 184 world-knowledge questions, referring to FELM-WK.

**Factcheck-Bench** (Wang et al., 2023) Factcheck-GPT gathered a total of 94 highly challenging questions from sources including Twitter posts, internal brainstorming, and Dolly-15k, encompassing 678 claims.

**FactScore-Bio** (Min et al., 2023) selected 183 entities, and collected responses from three LLMs including Davinci-text-003, ChatGPT, and PerplexityAI, and then annotated factual labels (supported, not-supported and irrelevant) for each atomic claim by humans.