Fact Check Technical Report: Cross-Validating Web Information via MITM Proxy and LLM Ensembles

Pang (Jeff) Liu

December 7, 2025

Artifact: `https://github.com/jeffliulab/fake_news_check`
Demo Video: `https://youtu.be/_5-A-WvDAkE`

Abstract

This project utilizes a custom **C-based HTTPS proxy** and a **multi-LLMs architecture** to build a mechanism for **cross-validating webpage information**. The system intercepts traffic to detect suspicious and false information in real-time. By combining a Man-In-The-Middle (MITM) proxy with a courtroom-style LLM ensemble, we achieved a 100% success rate in stability tests across 20 major websites. Notably, the Model Court module has been released as a python package **model-court** due to its scalability. This report provides the overall framework and technique information of our final project.

1 System Architecture

1.1 Proxy Design

The core connectivity component is a C-based HTTPS MITM proxy with optional LLM enhancement. On startup, it loads a CA certificate, initializes OpenSSL, creates a listening socket, and spawns a new thread for each client. Each thread determines the protocol based on the first request:

- **HTTP:** Handled by `handle_http_request`. It parses the Host, connects to the origin server, and forwards the request. In LLM mode, it buffers the full response and checks for uncompressed `text/html`. If suitable, the HTML body is sent to a local Flask server ('/enhance') as base64-encoded JSON. The proxy then rewrites the response with the enhanced body. If not enhanced, it falls back to the original response with the header `X-Proxy:CS112`.
- **HTTPS:** Handled by `handle_https_connect`. It parses `CONNECT host:port`, establishes a TLS connection to the real server, and sends `200 Connection Established`. Crucially, it generates a fake certificate signed by our CA to establish a TLS tunnel with the client. It then applies the same decryption and HTML enhancement logic as HTTP.

1.2 Parallel Processing Design

As shown in Figure 1, the system uses a parallel processing strategy to minimize latency. When the response arrives

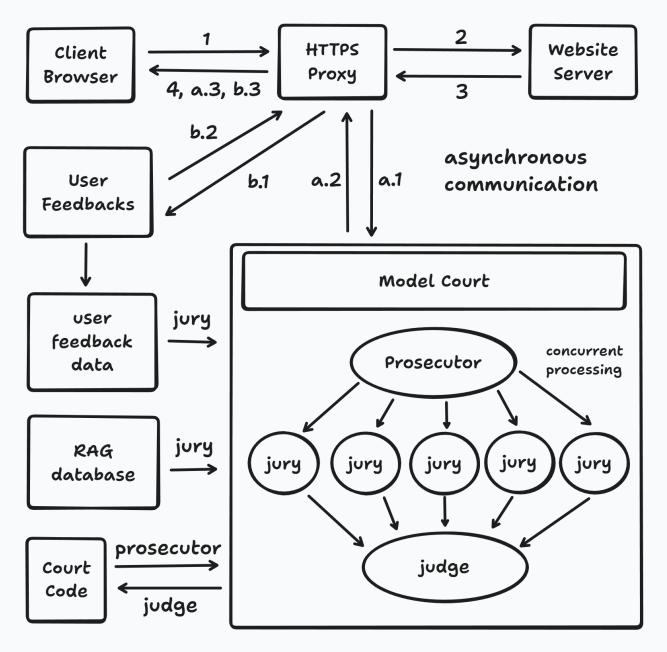

Figure 1: HTTPS Proxy and Enhancement Framework

Path 3, the proxy does three things simultaneously:

1. Streams the original page back to the browser immediately so the user can begin reading. (**Path 4**)
2. Sends page content to the Model Court agent. The agent will analyzed the content and give the judgment result, the user can view the page without receiving the final result. (**Path a.1/a.2**)
3. Event listening to if any user provide feedbacks and store locally. (**Path b.1/b.2**)

We create a reserved area on the top of the page. When Model Court finishes reasoning, the proxy injects the verdict into that area. We also provide a summary function that can summarize the page by parsing text content. This is a feature that reserved for future expansions.

1.3 Cross Validation via LLM Ensembles

We developed a **LLM ensembles** that can **cross validate the content with independent LLM models, and can be scalable on tools, RAG, AI agent, etc.** We separate published this module as a python package

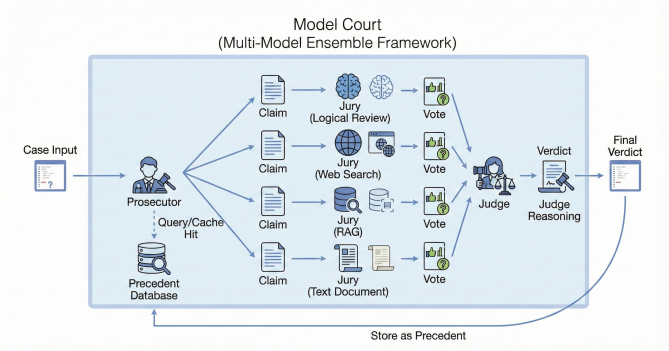

Figure 2: HTTPS Proxy and Enhancement Framework

model-court on PyPI, and will furthermore develop more functions as an open-source project. This idea comes from the movie ***12 Angry Men*** in 1957, so we use law terms to name critical variables.

The architecture operates in four distinct stages using an asynchronous pipeline to ensure low latency:

1. **The Prosecutor (Pre-processing):** The Prosecutor agent first analyzes the input case. It can employ an **Automatic Claim Splitting** mechanism to break complex statements into simple ones. Then Prosecutor will query the**Precedent Database**. If a semantically similar case is found, the system returns the cached verdict immediately, bypassing the expensive trial process.

2. **The Juries (Parallel Cross-Validation):** If a trial is necessary, the claim is dispatched to multiple independent juries running in parallel. To ensure robust cross-validation, we utilize five jurors implementations in our demo example, details can be seen at **Appendix B**. The juries must be independent to each other, and provide one of the following three opinions: no-objection, suspicious-fact, reasonable-doubt.

3. **The Judge (Verdict Aggregation):** The Judge aggregates jury opinions to determine the final verdict. While the aggregation logic is fully customizable, our demo implements the following standards: the verdict is labeled **CLEAN** if all jurors vote "no_objection"; if less than half of the jurors issue a non-"no_objection" vote, the verdict is **SUSPICIOUS**; and if half or more concur on an objection, it is **REFUTED**. A special exception applies: if there is exactly one non-"no_objection" vote, provided that the `RAG_Jury` maintains a "no_objection" stance, the verdict remains **CLEAN**. Final verdicts are persisted into the precedent database.

1.4 User Interaction

We also add a **user interaction mechanism** that user can give feedbacks if they think the analysis is not correct. They can provide their opionions and give their reasons. This is set as human-in-the-loop since user feedbacks are stored in text files and it can also be embedding into RAG if the amount of feedbacks are huge. Normally, the feedback should be added after correctness check. Verified human feedback will influence RAG-Jury and have a higher weight on voting. For example, in our demo, if the Judge receive one non-"no_objection" vote and User-Feedback-Jury give no-objection, then the overall verdict will be CLEAN.

2 Performance Evaluation

2.1 Quantitative Performance

We tested the proxy on 20 top global websites. The system achieved a **100% success rate**. The average overhead is approximately **287ms**, which is acceptable for real-time browsing. Fast sites like Microsoft responded in 115ms, while media-heavy sites like Netflix took 650ms. Summary statistics are in Table 1. Details see **Appendix A**.

Table 1: Proxy Latency Summary (n=20)

Metric	Value
Successful Tests	20 / 20
AVG Time	287.48 ms
MIN Time	115.41 ms
MAX Time	651.31 ms
STD Deviation	165.40 ms

2.2 Qualitative Results

We evaluated the fact-checking quality using several representative test cases. **(1)** For clean data, Wikipedia pages such as Tufts University were consistently identified as **CLEAN**, demonstrating reliable performance on trustworthy sources. **(2)** To assess robustness against misinformation, we created fake webpages claiming such as *"The US has sent a spaceship to Mars to build a colony."* In this case, the proxy successfully detected the content as an obvious fake fact and returned a **REFUTED** verdict. **(3)** For many politically news webpages, contents with strong narratives occasionally triggered **SUSPICIOUS** warnings, indicating potential bias without incorrectly classifying them as outright false.

3 Reflection and Future Work

Through this project, we can see the enormous potential of MITM proxy and cross-validation in practical use. In addition to the topics discussed above, we can also check for **harmful information, phishing information, and adult content** on web pages. Furthermore, by changing the direction of the inspection, we can also achieve corporate data security and prevent employees from illegally transmitting confidential information. **However**, people nowadays are increasingly prioritizing privacy and security, and **tunnel communication technologies** such as VPNs **cannot** be intercepted by MITM Proxy. At the same time, proxy interception also presents serious privacy issues. In the future, we plan to further expand the content of model-court and consider combining MITM Proxy with other technologies to achieve corporate data security.

A Appendix: Detailed Test Results

The following table lists the response time performance for each of the 20 tested websites.

Table 2: Per Site Performance Results

#	Website URL	Time (ms)
1	https://en.wikipedia.org/wiki/Tufts_University	141.26
2	https://www.google.com	224.70
3	https://www.youtube.com	318.65
4	https://www.github.com	359.21
5	https://www.amazon.com	135.03
6	https://www.twitter.com	592.87
7	https://www.facebook.com	194.72
8	https://www.instagram.com	435.22
9	https://www.reddit.com	362.67
10	https://www.cnn.com	165.27
11	https://www.bbc.com	118.42
12	https://www.nytimes.com	144.81
13	https://www.apple.com	160.00
14	https://www.microsoft.com	115.41
15	https://www.openai.com	125.54
16	https://www.netflix.com	651.31
17	https://stackoverflow.com	370.13
18	https://www.linkedin.com	518.82
19	https://www.cloudflare.com	363.66
20	https://www.ibm.com	251.82

B Multi-Jury Configuration of the Model Court System

Table 3 summarizes the configuration and functional roles of the five jurors used in the proposed Model Court fact-checking framework.

Table 3: Configuration of the Five Jurors in the Model Court System

ID	Name	Model	T	Function & Decision Rule
J1	Logic_GPT	openai gpt-4o-mini	0.0	Performs logical consistency and general knowledge verification, detecting contradictions, historical or physical impossibilities, and reasoning fallacies. Votes `no_objection` if the claim is logically sound and widely accepted as true.
J2	Logic_Gemini	google gemini-2.5-flash-lite	0.1	Acts as a Devil's Advocate for misinformation detection by identifying exaggerated, sensational, or manipulative framing. Votes objection immediately upon observing reasonable doubt.
J3	Web_Search_Jury	perplexity sonar	0.0	Conducts real-world verification using authoritative online sources such as news outlets, government websites, and encyclopedias. Votes `reasonable_doubt` if external evidence contradicts the claim with citation support.
J4	RAG_Jury	openai gpt-4o-mini	0.1	Verifies claims against a local administrator-maintained knowledge base using retrieval-augmented generation. Votes `suspicious_fact` if retrieved documents directly contradict the claim; abstains if context is irrelevant.
J5	User_Feedback_Jury	openai gpt-4o-mini	0.1	Performs community-driven historical verification based on previously reported corrections and fake claims stored in the trusted user feedback database. Votes according to verified community feedback records.