OpenScore: An Agent-Based Framework for Automated Evaluation of AI Model Transparency

Hyeon-Jun Han[1 [0009−0008−5129−0067]], Jae-Sang Jang[1 [0009−0000−2324−0013]], and Kang-Won Lee[1 [0000−0002−3025−4699]]

Sejong University, Seoul, 05006, Korea
{hyeonjun0119,dukejang01,kangwon.lee}@sju.ac.kr

Abstract. The openness of AI models is critical to the trust, reproducibility, and governance of AI systems. However, current methods for evaluating AI models are largely manual, subjective, inconsistent, and ineffective in keeping pace with the rapid release of new models. This paper presents **OpenScore**, the first end-to-end agent-based system for evaluating AI model openness. OpenScore automatically collects and analyzes evidence from various sources, including HuggingFace, GitHub, and arXiv, to assess models across well-defined criteria, spanning weights, code, license, data, and training methodology. Validated on well-known AI models, OpenScore achieves over 95% agreement with expert human evaluation while fully automating the entire evaluation process by agents. To support continuous transparency, we introduce a public leaderboard that will be updated automatically as new models appear. With its scalable and interpretable framework, OpenScore aims to provide the research community with clearer insights into the transparency of AI models.

Keywords: Openness of AI Models · Agent Architecture · AI Model Evaluation.

1 Introduction

Generative AI models, especially large language models (LLMs), are increasingly central to research, industry, and society. Their widespread deployment across diverse domains demands openness and transparency, which are critical for building trust, ensuring reproducibility, accelerating innovation, and meeting emerging regulatory requirements such as the EU AI Act and the Open Source Initiative's OSAID (Open Source AI Definition)[1]. Although many AI models are advertised as 'open', in practice they reveal only partial information and hides much about their internal mechanisms in a 'black box'. This issue has led to criticisms of so-called 'open-washing' practices, where openness is claimed without meaningful disclosure.

Current openness evaluations are largely conducted manually, requiring human experts to inspect GitHub repositories, model cards, research papers, and other documentations. Such manual efforts are slow, inconsistent, and cannot keep

pace with the rapid release of new AI models. As a result, the community lacks reliable, up-to-date visibility into model openness, making it difficult to track and compare transparency across models.

Several researchers have proposed frameworks for characterizing openness in the past. Liesenfeld et al. [2], Liu et al. [3], White et al. [4], Eiras et al. [5], and OSI's OSAID definition [1] outline useful criteria, but they stop short of offering operational, scalable solutions. Existing tools such as the Model Openness Tool (MOT) still require human input and do not scale effectively to new model releases. To date, there is no automated, continuous, and end-to-end system for evaluating AI model openness.

In this paper, we propose **OpenScore**, the first agent-based system for automated openness evaluation. OpenScore collects information from public sources such as HuggingFace, GitHub, arXiv, and blog posts. Specialized agents fetch and parse information across sixteen openness criteria—including weights, code, license, data, and training methods—based on the evaluation framework introduced in our prior work [6]. Each criterion is classified as Open, Semi-Open, or Closed, and aggregated into an overall openness score.

Our contributions are as follows:

1. We design and implement the first agent-based end-to-end automated system for evaluating AI model openness, integrating multiple public sources and assessing models using a structured framework.
2. We introduce a dynamic leaderboard that continuously updates as new models are released. This resource enables transparent, reproducible, and up-to-date comparison of openness across widely used AI models.
3. We evaluate the accuracy of our automated system by comparing it with human expert assessments, demonstrating strong alignment and accuracy with higher than 95% agreement.

From our evaluation, we find that the system scales efficiently in terms of execution time, compared to human experts. To promote transparency and reproducibility, we publicly release our agent source code so that other researchers can validate, extend, and build on our work. Through this effort, we aim to establish clearer guidelines for openness in AI, i.e., when researchers publish new models as 'open', there must be explicit evidence across well-defined criteria before such claims can be meaningfully substantiated.

2 Background and Related Work

Rapid adoption of generative AI models has created wide-ranging impacts across technology, science, education, government, and industry. With such broad influence, the openness of AI models has become an important concern. Openness enables scrutiny of technical development, reproducibility of research, acceleration of innovation, and trust in the responsible use of AI. Policymakers and

communities have voiced this importance, with initiatives such as the EU AI Act [2] and the Open Source Initiative's OSAID definition [1] pushing for more transparent and accountable AI practices. Without openness, AI risks concentrating power in the hands of a few, reducing accountability, and continuing 'black-box' decision making that hinders scientific progress and public trust.

At the same time, fully opening up AI models introduces a number of challenges. First, there are security concerns. When source code, model parameters, and other technical artifacts are made public, malicious users can misuse them for inappropriate or harmful purposes. Second, there are copyright and licensing issues. Open-source AI models vary in how their information can be used, redistributed, or modified. This may create legal uncertainty. Third, responsibility and liability remain unclear. When derivative models built on open releases cause harm, accountability is difficult to assign. Despite these risks, open models continue to gain traction due to their benefits for research and innovation.

Several frameworks have been proposed to characterize openness. Liesenfeld et al. [2] introduced a 14-item framework and criticized the practice of "openwashing," where limited disclosures are marketed as open source. Liu et al. [3] evaluated models on seven binary criteria, including weights and data sources, to promote more transparent release practices. Eiras et al. [5] combined popularity measures with a five-tier openness classification, while White et al. [4] proposed the Model Openness Framework (MOF), spanning open science, open tooling, and open models. The OSI's OSAID [1] provides a formalized definition across code, data, and models. Although these frameworks are valuable, they face two limitations: (i) the absence of a universally accepted definition of openness, and limited coverage of criteria, and (ii) reliance on manual assessment, which is slow, costly, and unsustainable.

Table 1. Openness Framework Coverage by Previous Research

	Model Basic Openness	**Accessibility & Reproducibility**	**Training Method Openness**	**Data Openness**
Ours [6]	◯	◯	◯	◯
OSAID [1]	◯	△	×	◯
Liesenfeld et al. [2]	◯	◯	×	◯
Liu et al. [3]	△	×	△	◯
Eiras et al. [5]	△	×	×	◯
White et al. [4]	◯	◯	×	◯

Our prior work addressed part of this issue by proposing a comprehensive, multidimensional openness evaluation framework [6]. We introduced criteria spanning model weights, code, license, data, training methods, accessibility, and reproducibility, and evaluated popular models using it. The evaluation criteria and selection guidelines for the system are detailed in the paper. Table 1 summarizes

the coverage of existing openness frameworks, where ◯ denotes at least 2/3 of criteria exist, × denotes no criteria exist, and △ denotes otherwise.

The advent of agent technologies provides an opportunity to resolve the bottleneck of manual evaluation. Agents can automate data collection from diverse sources and lightweight NLP and reasoning modules can parse documentation and classify models. By orchestrating specialized agents, the evaluation process becomes scalable, consistent, and capable of near-real-time updates. This paper builds on prior frameworks and moves beyond them by operationalizing openness evaluation using a fully automated agents.

3 The OpenScore System Architecture

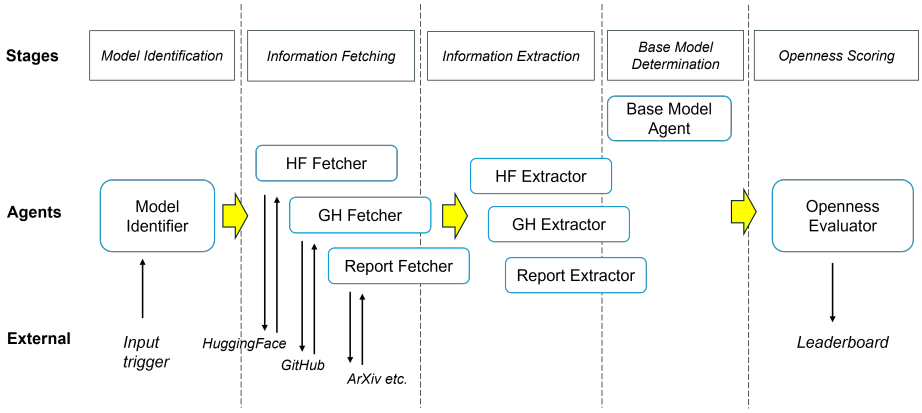

Fig. 1. Main Stages of the OpenScore System

Our agent-based evaluation architecture is designed to emulate the steps a human researcher would take when assessing the openness of an AI model. The system proceeds through five major stages: model identification, information fetching, information extraction and filtering, base model determination, and openness scoring. Figure 1 shows the main structure of this architecture.

3.1 Model Identification

The first step is to normalize the model identifier from a HuggingFace model ID or URL (e.g., `meta-llama/Llama-3-8B`). Using this identifier, the system searches for the corresponding GiHub repository and technical papers. One of the challenges is that the HuggingFace entry may not contain explicit links about the corresponding GitHub or tech reports. So we first identify candidate repositories. Each candidate is scored based on several signals:

- Organization affinity: similarity between repository organization and HuggingFace organization.
- Token hits: presence of model-specific token (e.g., model name, size, version) in repository metadata.
- Explicit link bonus: preference for repositories that are explicitly linked on HuggingFace.
- Penalties: applied for misleading and unqualified keywords or mismatching version numbers.

A candidate is accepted if its score exceeds some threshold. We present more details about this stage in Section 4.

3.2 Information Fetching

Once repositories and references are identified, the system utilizes specialized fetcher agents to collect model information from three key sources:

- HuggingFace Fetcher: collects model metadata, configuration, weights, tokenizer and documentation.
- GitHub Fetcher: gathers implementation details of the model and supplementary documents from the selected repository.
- Report Fetcher: retrieves academic papers of technical reports describing the model's training, data, and methodology from arXiv, blog posts, Web sites, and other sources.

In this stage, the job of information fetching agents is similar to crawling. These agents work in parallel to collect the data that they are tasked to gather and store the information for later consumption.

3.3 Information Extraction and Summarization

Fetched documents are processed by extractor agents that analyze the content. The goal of the analysis is to determine the openness level of the model based on the criteria defined in our evaluation framework [6]:

- Model basic openness: evaluates the openness of weights, code, license, paper, architecture, tokenizer.
- Accessibility and reproducibility: evaluates information about hardware, software used for model training, and availability of APIs.
- Training methodology openness: evaluates information about pre-training, fine-tuning, reinforcement learning methods.
- Data openness: evaluates availability of datasets used for pre-training, fine-tuning, reinforcement learning, and filtering.

For each criterion, the system extracts evidences (from web pages and technical documents) and generates concise summaries. Fine-tuning and reinforcement learning are carefully studied and classified as *used, not used*, or *unknown* based on the evidence whether these methods were used for model generation or not.

3.4 Base Model Determination

Fine-tuned or instruction-tuned models often inherit from a base model. To ensure completeness, the system identifies and evaluates the pre-training model by analyzing metadata for references such as "`based_model`" or "`fine-tuned from`" on the HuggingFace entry. If no explicit reference is found, heuristics and LLM-based text analysis are used to further investigate. Identified base models are evaluated on training methodology and data openness as they provide grounds for the basic methods, and their results are used to evaluate the derived models.

3.5 Openness Evaluation and Scoring

The extracted evidence is aggregated into the final openness score. Each criterion is classified as Open, Semi-Open, or Closed based on the evaluation. The final score is normalized to the 0 – 10 range:

$$Openness\ Score = \frac{1}{16} \sum_{i=1}^{16} w_i \times \text{score}\ (criterion_i) \times 10 \tag{1}$$

where w_i represents the weight of each criterion and *Open, Semi-Open*, and *Closed* are assigned 1.0, 0.5, and 0.0, respectively. In this work we set $w_i = 1$ treating all criteria equal, while leaving potential for weighted scoring as we gain more insights. If a model is completely open the score is 10. If it is completely closed the score is 0. The criterion-level scores are preserved alongside the aggregate score. This ensures both transparency and interpretability of the evaluation results.

3.6 Leaderboard Generation

The final stage of the process is constructing the leaderboard to make the openness results available to the public. The system stores openness scores, criterion-level evaluations, and supporting evidence in a central database. The leaderboard displays each model's aggregate score, per-criterion openness levels, short text showing evidence, and metadata such as organization, release date, and the base model. Models are ranked to enable transparent comparison among AI models, and the results get updated continuously as new models appear.

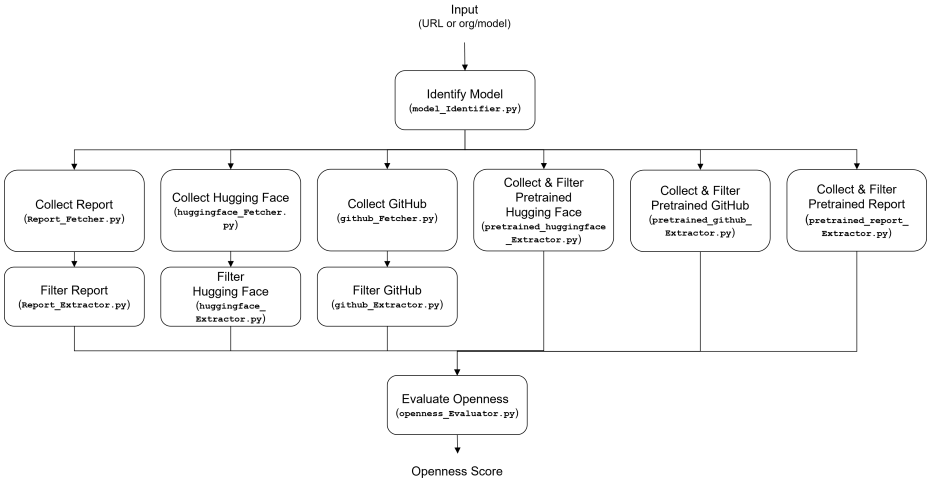

Fig. 2. End-to-End Pipeline for Agent-Based Openness Evaluation

4 Implementation

We implemented the proposed architecture in Python, using a modular agent-based design pattern. Figure 2 illustrates the overall workflow of the proposed architecture. Each stage is implemented by a set of specialized agents and supporting utilities. The implementation was designed with explainability in mind: intermediate results are serialized to JSON and stored, and every component produces outputs that can be inspected. Our code is available at `https://github.com/sejong-sas/Evaluating-system-based-on-agent`.

4.1 Model Identification

One of the challenges in implementing the system is that HuggingFace entries are not consistent across models. Some models have all necessary information on the HuggingFace page, while others have distributed information sources (e.g., GitHub, arXiv, blog post, etc.) that need to be matched first in order to accurately evaluate the openness of the model.

Candidate GitHub Repositories If the HuggingFace model card or README file contains official GitHub links, the system extracts them and records the corresponding organizations in an `allowed_org` set. For example, the HuggingFace model `bigscience/bloom` would generate `bigscience` as an allowed organization.

If no repository link is identified, we use Tavily Web Search API and GitHub search API, querying search keywords such as "`[model name] official GitHub repository`." The top results are then considered as candidate repositories.

Repository Scoring Each candidate repository is scored using a weighted function that incorporates four signals.

- **Organization Affinity (α)**: If the organization name explicitly appears in the HuggingFace model card or is a member of the `allowed_org` set, a weight α_1 is assigned. If the organization is not explicitly linked but shows strong textual similarity to the name on HuggingFace, a weight $\alpha_2 (< \alpha_1)$ is assigned. Otherwise, a penalty $-\alpha_3$ is applied.
- **Token Hits (β)**: It measures if the repository contains the model name. If tokens derived from the model ID appear in the repository name, a weight β_1 is assigned. Occurrences of tokens in the README or other files add a weight β_2 and β_3, respectively.
- **Explicit Link Bonus (γ)**: Repositories discovered directly through HuggingFace links receive a bonus weight γ, while the ones identified via Web or GitHub search do not.
- **Penalties (δ)**: Repositories that include unqualified keywords (e.g., `ggml`, `demo`) receive a penalty δ_1 applied per occurrence up to a defined cap. Repositories that contain incorrect version references (e.g., `llama-2` when evaluating Llama-3) incur an immediate penalty δ_2.

The final repository score is computed as a weighted sum $S = \sum_i^3 \alpha_i + \sum_i^3 \beta_i + \gamma - \sum_i^2 \delta_i$.

Repository Selection The agent selects a repository if its score exceeds a threshold τ, with different threshold values depending on whether the organization belongs to the `allowed_org` set or not. If the repository organization belongs to `allowed_orgs` then a lower threshold is applied. Otherwise, stronger evidence is required: tokens must appear in both the repository name and README/file paths, and a higher threshold is used. Among valid candidates, the repository with the highest score is chosen. If none of them meets the threshold, the result is set to "none". We note that the parameters used for scoring and the threshold τ are empirically decided. We summarize the parameters in Table 2.

Table 2. Scoring Parameters

Parameters	Description	Value
$\alpha_1, \alpha_2, \alpha_3$	Organization affiliation	9, 6, -6
$\beta_1, \beta_2, \beta_3$	Token hits (per occurrence up to max)	6, 3 (max 6), 2 (max 4)
γ	Explicit link bonus	5
δ_1, δ_2	Penalties (per occurrence up to max)	2 (max 8), 8
τ	Selection threshold	4 (low), 8 (high)

4.2 Information Fetching

Given a model identifier, the system collects information from multiple sources. This process is handled by specialized fetchers: HuggingFace Fetcher, GitHub Fetcher, and Report Fetcher. They operate in parallel following the parallel agent pattern.

HuggingFace Fetcher The HuggingFace Fetcher reads the HuggingFace page and extracts metadata and repository information. More specifically, it collects `README.md, config.json, generation_config.json, LICENSE`, and all .py files. To build evidence the agent collects quotes that represent each openness criterion from the relevant source files. The results are serialized into the intermediate output called `huggingface_modelID.json`. In addition, the agent looks for external links to technical reports or blog posts. The documents are parsed, and their text content is saved to `reports_fulltext_huggingface_modelID.json`.

GitHub Fetcher The GitHub Fetcher targets the repository found in the model identification stage. Similar to the HuggingFace Fetcher, it extracts `README, LICENSE`, and all `.py` source files in the GitHub repository, and stores evidence from them in `github_repo.json`. It also scans linked reports and blogs, and analyzes them to generate evidence from these sources. This result is saved in `reports_fulltext_github_repo.json`.

Finally, the reports generated by the HuggingFace and GitHub Fetchers are merged into `reports_fulltext_modelID.json`.

Report Fetcher The Report Fetcher identifies papers or technical reports associated with the model from arXiv repository or blog posts. If a HuggingFace page provides an arXiv link, the PDF is downloaded and processed for text extraction (using PyMuPDF). If no arXiv link is found, we search the Web to identify candidate papers. The content of the paper is analyzed (using OpenAI API) to ensure that the content is indeed about the model. The results are serialized to `arxiv_fulltext_model_ID.json`.

4.3 Information Extraction and Summarization

After the information fetching step, the system processes the outputs from the three agents using extractor agents, namely HuggingFace Extractor, GitHub Extractor and Report Extractor. These agents map the collected output onto the 16 openness criteria of our evaluation framework [6]:

- Model basic openness (6 criteria): weights, code, license, paper, architecture, tokenizer
- Accessibility and reproducibility (3 criteria): hardware, software information, API availability

- Training methodology (3 criteria): pre-training, fine-tuning, reinforcement learning methods
- Data openness (4 criteria): information about datasets used for pre-training, fine-tuning, reinforcement learning, and data filtering

For each criterion, the extractors identify potentially relevant sentences along with their sources. These evidences are used for evaluation, and then summarized into concise statements, resulting in an evidence structure composed of *summary, quote*, and *source*.

Evidence related to fine-tuning and reinforcement learning is further analyzed with targeted prompts to determine whether such methods were actually applied, leading to categorical labels (*used, not-used, unknown*).

4.4 Base Model Determination

Many AI models inherit from base models (e.g., `Llama-3.1-8B-Instruct` is derived from `Llama-3.1-8B`). Thus we aim to resolve such dependencies when evaluating the openness of a model. To do that the system scans HuggingFace `README`s and model cards for patterns such as `"base_model:"` `"fine-tuned from"`. Candidate models are then verified via a HuggingFace API lookup. If explicit references are not found, heuristics are used to strip suffixes (e.g., `instruct, chat, sft`) to search for base models. As the final method, the system uses LLM queries to identify the most plausible candidates.

When a pre-training model is identified, we use the same types of extractors (for HuggingFace, GitHub, and Report) except that we employ a reduced pipeline – limited to training methodology and training data openness – is executed, and the results are merged with the derived model's evidence.

4.5 Openness Evaluation

The final stage of the architecture is the Openness Evaluator, which integrates evidence from all extractors, including the ones of the base model determination. Each criterion is classified into one of the three states: *Open* (1.0), *Semi-Open* (0.5), or *Closed* (0.0). We guide the classification using a combination of algorithmic checks and outputs from the language model to reduce errors that can be introduced by ambiguity or even hallucination. For example, the presence of `.safetensors` or `.bin` files is a clear indication of open weights. Similarly the presence of `tokenizer.json` or `vocab.json` sets the tokenizer criterion to be Open. These algorithmic checks override or refine LLM judgments, ensuring that clear-cut cases are scored consistently.

Certain criteria require additional safeguards. For example, when API availability is not explicitly documented in HuggingFace, GitHub, or associated papers – as is the case for models such as Mixtral and Gemma – the system issues a

Web-enabled query using GPT-4.1-mini to confirm whether a public API exists or not.

The final score is computed using the formula presented in Section 3.5. This evaluation yields both an interpretable aggregate measure of openness and a per-criterion breakdown that preserves transparency for downstream analysis.

4.6 Leaderboard Generation

All evaluation outputs are consolidated into a leaderboard database. Each entry includes the model identifier, organization, release date, base model (if applicable), the aggregate openness score, and the per-criterion classification with supporting evidence. A Web interface allows users to search, filter, and compare models. The system re-runs incrementally as new models are released or documentation changes, ensuring that the leaderboard remains current without manual intervention.

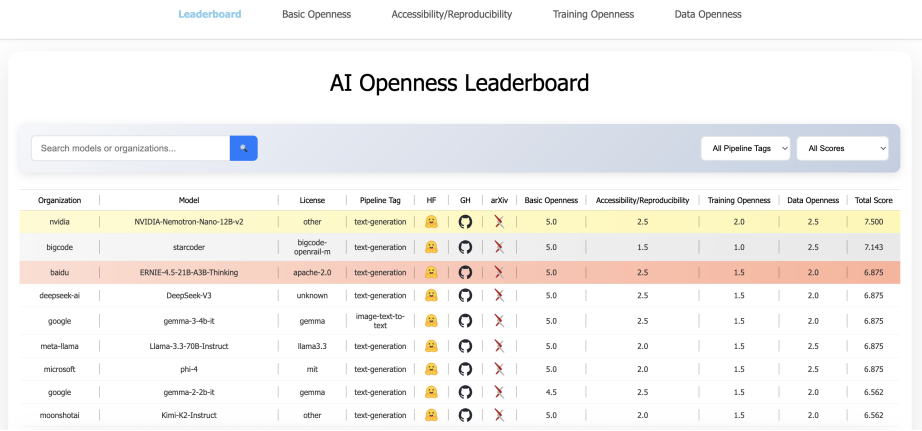

Fig. 3. The AI Openness Leaderboard Screenshot

Figure 3 presents a screenshot of the leaderboard. The leaderboard displays a model's openness score and related metadata on the UI based on the given model input. The leaderboard supports a search function based on organization and model name, as well as sorting and filtering function by pipeline tag and score classifications. For more detailed views of the openness evaluation, users can also check the breakdown of individual scores for each openness metric.

5 Evaluation

We evaluate the proposed system using 29 AI models. We categorize them into two groups: Group 1 consists of 19 LLM models that were studied in our previous

Table 3. Openness Scores for Group 1 by Agent and Human (in parentheses)

ORG	Model Name	Model Basic Openness	Accessibility & Reproducibility	Training Methodology Openness	Data Openness	Total Score
Google DeepMind	Gemma-3 [7]	5(5)	2.5(2.5)	1.5(1.5)	2(2)	6.88(6.88)
Google DeepMind	Gemma-2 [8]	5(5)	2.5(2.5)	1.5(1.5)	2(2)	6.88(6.88)
Qwen AI	QwQ [9]	4.5(4.5)	0(0)	0(0)	0.5(0.5)	3.13(3.13)
Qwen AI	Qwen2.5 [10]	5(5)	1(1)	1.5(1.5)	2.5(2.5)	6.25(6.25)
DeepSeek AI	DeepSeek R1 [11]	5(5)	1(1)	1.5(1.5)	2(2)	5.94(5.94)
DeepSeek AI	DeepSeek V3 [12]	5(5)	2.5(2.5)	1.5(1.5)	2(2)	6.88(6.88)
Mistral AI	Mistral-small-3.1 [13]	4.5(4.5)	1(1)	0(0)	0(0)	3.44(3.44)
Meta AI	Llama 3.3 [14]	5(5)	2.5(2.5)	1.5(1.5)	2.5(2.5)	7.19(7.19)
Microsoft	Phi-4 [15]	5(5)	2(2)	1.5(1.5)	2.5(2.5)	6.88(6.88)
Cohere	c4ai-command-a [16]	4.5(4.5)	1(1)	1.5(1.5)	2(2)	5.63(5.63)
Mistral AI	Mixtral [17]	5(5)	0(0)	1.5(1.5)	1.5(1.5)	5(5)
OpenAI	GPT-2 [18]	5(5)	1(1)	0.5(0.5)	1(1)	4.69(4.69)
BigCode	Starcoder [19]	5(5)	1.5(1.5)	1(1)	2.5(2.5)	6.25(6.25)
Meta AI	OPT [20]	4.5(4.5)	1.5(1.5)	0.5(0.5)	1(1)	4.69(4.69)
TII	Falcon3 [21]	4.5(4.5)	1(1)	1(1)	1(1)	4.69(4.69)
BigScience	BloomZ [22]	**5(6)**	2(2)	**0.5(2)**	**1(2)**	**5.31(7.5)**
LG	EXAONE-Deep [23]	4.5(4.5)	1(1)	1(1)	1.5(1.5)	5(5)
Kakao	Kanana [24]	4.5(4.5)	0.5(0.5)	1.5(1.5)	1.5(1.5)	5(5)
Naver	HyperCLOVA X [25]	5(5)	1.5(1.5)	1.5(1.5)	2.5(2.5)	6.56(6.56)

Table 4. Openness Scores for Group 2 by Agent and Human (in parentheses)

ORG	Model Name	Model Basic Openness	Accessibility & Reproducibility	Training Methodology Openness	Data Openness	Total Score
xAI	grok-2 [26]	3.5(3.5)	1(1)	**0.5(0)**	0(0)	**3.13(2.81)**
OpenAI	gpt-oss [27]	5(5)	1.5(1.5)	0.5(0.5)	1.5(1.5)	5.31(5.31)
NVIDIA	NVIDIA-Nemotron-Nano [28]	5(5)	2.5(2.5)	**2(1.5)**	2.5(2.5)	**7.5(7.19)**
Baidu	ERNIE-4.5 [29]	5(5)	2.5(2.5)	1.5(1.5)	2(2)	6.88(6.88)
Moonshot AI	Kimi-K2-Instruct [30]	5(5)	2(2)	1.5(1.5)	2(2)	6.56(6.56)
Tencent	Hunyuan-A13B [31]	4.5(4.5)	1(1)	1.5(1.5)	2(2)	5.63(5.63)
Line-Corporation	japanese-large-lm [32]	4.5(4.5)	**0(0.5)**	**0.5(0)**	0.5(0.5)	**3.44(3.44)**
CyberAgent	open-calm [33]	4(4.5)	0.5(0.5)	0.5(0.5)	0.5(0.5)	3.44(3.44)
LLM-jp	llm-jp-13b-instruct [34]	5(5)	2(2)	**1.5(1)**	1.5(1.5)	**6.25(5.94)**
Tokyotech-llm	Llama-3-Swallow [35]	4.5(4.5)	0.5(0.5)	1(1)	1.5(1.5)	4.69(4.69)

work [6]. Group 2 consists of 10 new models including Grok, NVIDIA Nemotron, Hunyuan, and Line LM from U.S., China, and Japan. We note that the openness results from the first group, which was reported in [6], were used to develop and fine-tune our agent-based automated evaluation system. We then used the second group to test the performance of the agent system. In this regard, Group 1 serves as a training set for our system and Group 2 serves as a test set.

Table 3 shows the openness scores of Group 1 calculated by human experts and AI agents. We mark the cases when the results from human and AI do not agree with bold face. The scores computed by the agent system are shown next to the scores computed by human (in parentheses). In the table there is only one case, namely BloomZ from BigScience, that has differences in scores. We noticed that in the BloomZ case, information about the training code resides in three

different files[1]. But the agent stopped searching after identifying the first file[2], hence its scores for code, training methodology, training data, and data filtering were inaccurate.

Table 4 presents the openness scores of Group 2, again calculated by AI agents and human experts. Again we mark the cases with differing results with bold face. In the table, we observe a few cases where the results by human expert and AI agent disagree. This stems from practical difficulties in developing an agent-based model evaluation system. For example, in the case of grok-2, in the arXiv fetcher stage, a Web search incorrectly returned a paper titled "Language Model Grok to Copy", which is not related to the model grok, but about a practice called "grokking". Although the final openness evaluation of that paper was classified as closed, content from the paper remained in the extraction step, which ultimately led the agent to label the pre-training methodology openness as semi-open.

To evaluate the performance of the agent system, we compute the accuracy and agreement of the system compared with human experts. For this, we use micro f1-score and Gwet's AC1.[3] They are computed as follows.

Let $K = 3$ unordered categories (*Open*, *Semi-Open*, *Closed*). Let n_{ij} be the number of items where the human (row index i) chose category $i \in \{1, \ldots, K\}$ and the agent (column index j) chose category $j \in \{1, \ldots, K\}$. Define

$$N = \sum_{i=1}^{K} \sum_{j=1}^{K} n_{ij}, \qquad n_{i\cdot} = \sum_{j=1}^{K} n_{ij}, \qquad n_{\cdot j} = \sum_{i=1}^{K} n_{ij}.$$

Micro F1-score: For single-label multi-class tasks, the micro F1-score equals accuracy:

$$\text{Micro F1} \;=\; \frac{\sum_{i=1}^{K} n_{ii}}{N}. \tag{2}$$

Gwet's AC1: Let the pooled (averaged) marginal proportion for category i be

$$p_i \;=\; \frac{n_{i\cdot} + n_{\cdot i}}{2N} \quad (i = 1, \ldots, K). \tag{3}$$

Then the observed agreement A_o and the chance agreement A_e are

$$A_o \;=\; \frac{\sum_{i=1}^{K} n_{ii}}{N}, \qquad A_e \;=\; \sum_{i=1}^{K} p_i\big(1 - p_i\big) \;=\; 1 - \sum_{i=1}^{K} p_i^2. \tag{4}$$

[1] `repositories-bigscience-workshop/xmtf, bigscience-workshop/bigscience, bigscience-workshop/Megatron-DeepSpeed`

[2] `repositories-bigscience-workshop/xmtf`

[3] Although Cohen's κ can be used to measure agreement, it is known to suffer from the *prevalence/bias paradox*—when the marginal distributions are highly imbalanced, κ can yield deceptively low agreement even under near-perfect concordance. Gwet's AC1 [36] mitigates this issue by redefining the chance agreement using the pooled (averaged) marginals.

Table 5. Similarity evaluation: F1-score (micro) and Gwet's AC1 by group

Similarity Evaluation	F1-score (micro)		Gwet's AC1	
	Group 1	Group 2	Group 1	Group 2
Weight	1.00	1.00	1.00	1.00
Code	**0.95**	**1.00**	**0.94**	**1.00**
License	1.00	1.00	1.00	1.00
Paper	1.00	1.00	1.00	1.00
Architecture	1.00	1.00	1.00	1.00
Tokenizer	1.00	1.00	1.00	1.00
Hardware	**1.00**	**0.90**	**1.00**	**0.74**
Software	1.00	1.00	1.00	1.00
API	1.00	1.00	1.00	1.00
Pre-training	**0.95**	**0.70**	**0.93**	**0.59**
Fine-tuning	**0.95**	**1.00**	**0.92**	**1.00**
RL/DPO	**1.00**	**0.90**	**1.00**	**0.80**
Pre-training Data	1.00	1.00	1.00	1.00
Fine-tuning Data	1.00	1.00	1.00	1.00
RL/DPO Data	1.00	1.00	1.00	1.00
Data Filtering	**0.95**	**1.00**	**0.86**	**1.00**
Average	**0.99**	**0.97**	**0.98**	**0.95**

Finally,

$$\text{AC1} = \frac{A_o - A_e}{1 - A_e}. \tag{5}$$

Table 5 presents the *micro F1-score (average across items)* and *Gwet's AC1* [36] so that we can understand the performance of agent system and agreement between human results and agent result. As shown in Table 5, Group 1 exhibits uniformly high similarity (avg. F1 = 0.99, AC1 = 0.98), while Group 2 remains strong overall but slightly lower on average (avg. F1 = 0.97, AC1 = 0.95). The most notable declines occur in pre-training, hardware, and RL/DPO.

These gaps primarily stem from: (i) mis-fetched literature (e.g., incorrect arXiv hits); (ii) version recognition errors; and (iii) the difficulty of faithfully instantiating the human grading rubric in the agent. Given the same evidence, human raters may judge it sufficiently explicit using context, whereas the agent, relying more on predefined quotations and pattern rules, tends to classify conservatively, producing divergences.

Despite these challenges, our proposed agent system produces highly accurate openness results across wide variety of AI models. We plan to further refine the agent system so that it can handle wide variety of AI models.

Table 6 summarizes the minimum, average, maximum, and standard deviation of the agent's execution time from model identification to openness evaluation

Table 6. End-to-End Execution Time for Evaluation by Agents

Min	Avg	Max	Std
7.73 min	48.68 min	178.56 min	38.36 min

Table 7. Execution Time by Agents and Human Raters (Meta AI's OPT)

Agent	Human 1		Human 2	
Time	Time	Speed Up	Time	Speed Up
46.24 min	278 min	6.01×	298 min	6.44×

across the combined set of 29 models (Group 1 + Group 2). Baidu's ERNIE-4.5 required the shortest time (7.73 min), while Google DeepMind's Gemma-3 took the longest (178.56 min) for the OpenScore system to evaluate their openness.

Table 7 reports the comparison of the time taken by two human experts to assess the openness of a model alongside the agent's time for the same task. We selected Meta AI's OPT model since its agent execution time is similar to the average of the overall agent execution time. The time for human experts was measured for just the time that were taken by the human raters on reading documents and making decisions without rest. For this model, the agent took 46.24 min to complete the evaluation, while the manual evaluation by the first human expert took 278 min and that the second human expert took 298 min, corresponding to speedups of 6.01× and 6.44×, respectively.

6 Conclusion

This paper introduced the first end-to-end agent-based system for automated evaluation of AI model openness. By combining structured evidence gathering from HuggingFace, GitHub, and arXiv with a multi-stage extraction and scoring pipeline, our approach replicates the workflow of human researchers while scaling to a much larger number of models in the same time duration. The system integrates 16 established openness criteria and produces interpretable per-criterion results. The resulting openness score enables consistent and transparent comparison across models. Finally, the continuously updated leaderboard further promotes the community's awareness of the openness of AI models.

Preliminary evaluation shows that the automated assessments closely align with expert human judgement, while allowing speed, scalability, and consistency. Moreover, the leaderboard may disclose discrepancies between perceived openness versus actual evidence of openness in popular AI models, shedding light on potential 'open-washing'.

Going forward, we plan to make our agent-based evaluation system and leaderboard publicly accessible to encourage participation from the community. This will allow us to refine the scoring framework based on community input and engagement. Furthermore, we plan to incorporate meaningful case studies and user feedback derived from this participation to further enhance the framework. We also aim to explore collaborations with other policy initiatives, such as OSAID, so that this important issue can be addressed through a concerted effort.

Acknowledgements

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the National Program for Excellence in SW, supervised by the IITP (Institute of Information Communications Technology Planning & Evaluation) in 2025 (2024-0-00037).

References

1. Open Source AI Definition (OSAID) 1.0, Open Source Initiative (OSI) Homepage, `https://opensource.org/ai/open-source-ai-definition`.
2. A. Liesenfeld, M. Dingemanse: Rethinking open-source generative AI: open-washing and the EU AI Act. In: ACM Conference on Fairness, Accountability, and Transparency (FAccT). ACM, 2024.
3. Z. Liu et al.: LLM360: Towards fully transparent open-source LLMs. arXiv preprint arXiv:2312.06550 (2023).
4. M. White et al.: The Model Openness Framework: Promoting completeness and openness for reproducibility, transparency, and usability in artificial intelligence. arXiv preprint arXiv:2403.13784 (2024).
5. F. Eiras et al.: Risks and opportunities of open-source generative AI. arXiv preprint arXiv:2405.08597 (2024).
6. K.-W. Jeon, H.-J. Han, K.-W. Lee: Evaluating the openness of impactful AI models with a focus on LLMs. The Transactions of the Korea Information Processing Society (TKIPS) 14(6), 468–479 (2025).
7. Gemma Team: Gemma 3 Technical Report. arXiv:2503.19786 (2025).
8. Gemma Team: Gemma 2: Improving Open Language Models at a Practical Size. arXiv:2408.00118 (2024).
9. Qwen Team: QwQ-32B (reasoning model) — model card (online), `https://huggingface.co/Qwen/QwQ-32B`, last accessed 2025/09/20.
10. Qwen Team: Qwen2.5 Technical Report. arXiv:2412.15115 (2024).
11. DeepSeek-AI: DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 (2025).
12. DeepSeek-AI: DeepSeek-V3 Technical Report. arXiv:2412.19437 (2024).
13. Mistral AI: Mistral Small 3.1 announcement (blog, online), `https://mistral.ai/news/mistral-small-3-1`, last accessed 2025/09/20
14. Meta AI: The Llama 3 Herd of Models. arXiv:2407.21783 (2024).
15. Abdin, M. et al.: Phi-4 Technical Report. arXiv:2412.08905 (2024).
16. Cohere Labs: Command A: An Enterprise-Ready Large Language Model. arXiv:2504.00698 (2025).

17. Mistral AI: Mixtral of Experts. arXiv:2401.04088 (2024).
18. Radford, A. et al.: Language Models are Unsupervised Multitask Learners. OpenAI report (2019).
19. Li, R. et al.: StarCoder: may the source be with you! arXiv:2305.06161 (2023).
20. Zhang, S. et al.: OPT: Open Pre-trained Transformer Language Models. arXiv:2205.01068 (2022).
21. TII: Falcon 3 overview (online), `https://falconllm.tii.ae/falcon3/index.html`, last accessed 2025/09/20.
22. Muennighoff, N. et al.: Crosslingual Generalization through Multitask Finetuning (BLOOMZ). arXiv:2211.01786 (2022).
23. LG AI Research: EXAONE Deep: Reasoning Enhanced Language Models. arXiv:2503.12524 (2025).
24. Kanana LLM Team (Kakao): Kanana: Compute-efficient Bilingual Language Models. arXiv:2502.18934 (2025).
25. Naver HyperCLOVA X Team: HyperCLOVA X THINK Technical Report. arXiv:2506.22403 (2025).
26. xAI: Grok-2 beta release (blog, online), `https://x.ai/news/grok-2`, last accessed 2025/09/20.
27. OpenAI: Introducing gpt-oss (blog, online), `https://openai.com/index/introducing-gpt-oss/`, last accessed 2025/09/20.
28. NVIDIA Research: NVIDIA Nemotron-Nano-2 Technical Report. arXiv:2508.14444 (2025) / PDF TR (online), last accessed 2025/09/20.
29. Baidu: Announcing the Open-Source Release of ERNIE 4.5 (blog, online), `https://yiyan.baidu.com/blog/posts/ernie4.5`, last accessed 2025/09/20.
30. Moonshot AI: Kimi-K2 (project page, online), `https://moonshotai.github.io/Kimi-K2/`, last accessed 2025/09/20.
31. Tencent: Hunyuan-A13B-Instruct — model card (online), `https://huggingface.co/tencent/Hunyuan-A13B-Instruct`, last accessed 2025/09/20.
32. LINE Corporation: japanese-large-lm-3.6b — model card (online), `https://huggingface.co/line-corporation/japanese-large-lm-3.6b`, last accessed 2025/09/20.
33. CyberAgent: OpenCALM-7B — model card (online), `https://huggingface.co/cyberagent/open-calm-7b`, last accessed 2025/09/20.
34. LLM-jp Consortium: LLM-jp — project paper overview. arXiv:2407.03963 (2024).
35. TokyoTech-LLM & Swallow Team: Llama 3 Swallow (project page, online), `https://swallow-llm.github.io/llama3-swallow.en.html`, last accessed 2025/09/20.
36. Gwet, K. L.: Handbook of Inter-Rater Reliability: The Definitive Guide to Measuring the Extent of Agreement Among Raters. 4th edn. Advanced Analytics, Gaithersburg (2014).