The API Privacy Paradox

How LLM APIs Unintentionally Reveal Model Details

Sean Ren | Associate Professor, USC

xiangren@usc.edu | @xiangrennlp

LLMs are often deployed behind API endpoints to keep the model's architecture and weights secret.

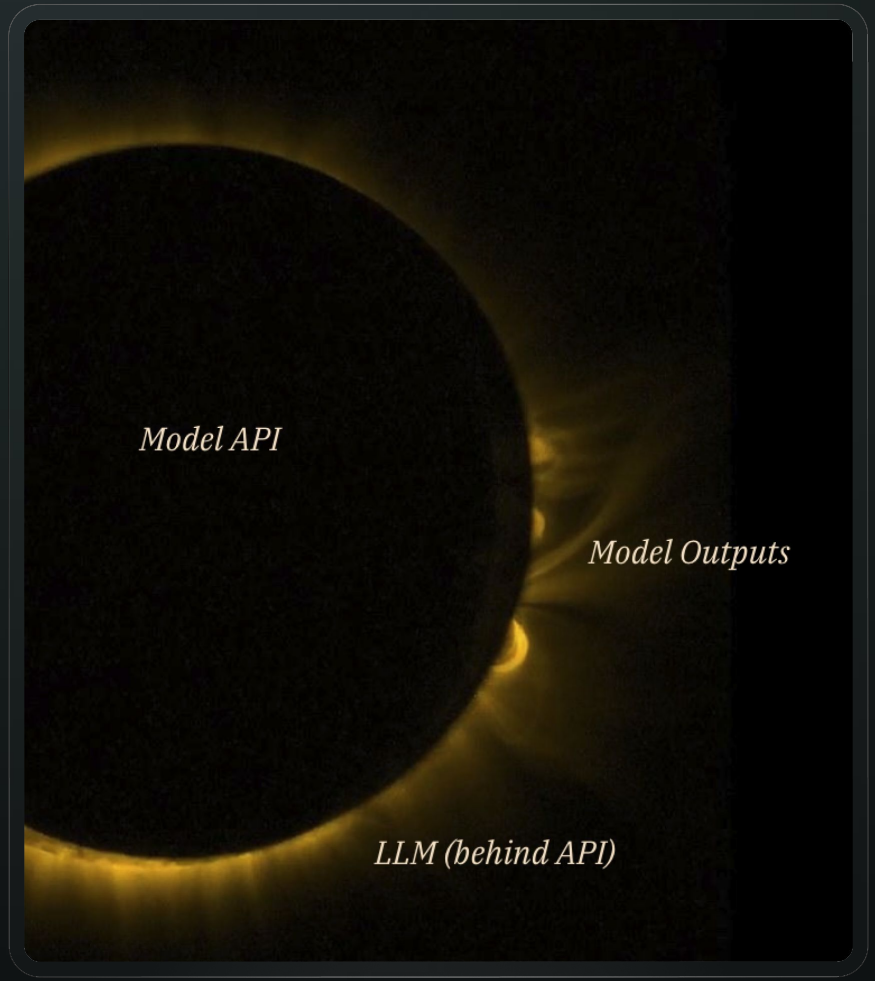
The assumption is that limited API access protects proprietary details... But is this true?

System outputs reflect underlying structure

During an eclipse, scientists observe the structure of the corona (solar atmosphere) to study the magnetic topography of the temporarily hidden sun.

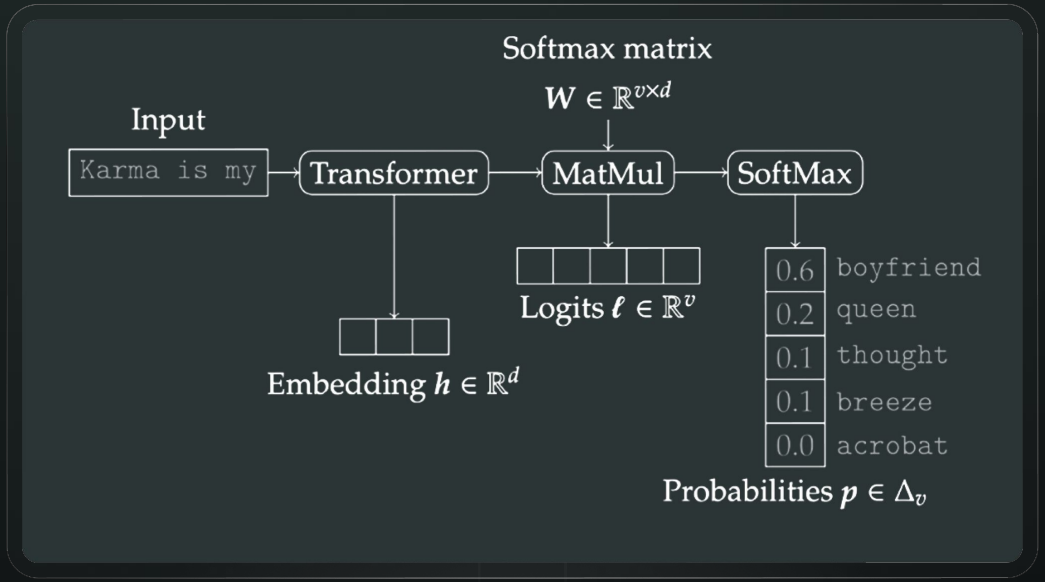
What do LLM APIs Reveal?

- APIs don't just return text outputs; they can also return probabilities/logits.
- Probability distributions embed subtle architectural signals
- The Result: False sense of security as outputs can reveal hidden details



Language Models

- When you give an LLM a prompt, it assigns a score (called a logit) to every possible next word
- These scores come from a fixed-size hidden state (embedding h)



Language Models *Secrets*

2 Scope and Limitations of this Technical Report

This report focuses on the capabilities, limitations, and safety properties of GPT-4. GPT-4 is a Transformer-style model [39] pre-trained to predict the next token in a document, using both publicly available data (such as internet data) and data licensed from third-party providers. The model was then fine-tuned using Reinforcement Learning from Human Feedback (RLHF) [40]. Given both the competitive landscape and the safety implications of large-scale models like GPT-4, this report contains no further details about the architecture (including model size), hardware, training compute, dataset construction, training method, or similar.

Companies like OpenAI hide model details to protect their business.

Secrets include model parameters like the LM head and hyperparameters like the embedding size.

GPT-4 Technical Report
https://arxiv.org/abs/2303.08774

API Assumptions

The mainstream APIs give top-k logprobs where k is usually 5:

```
TOP-K API OUTPUT
{
    "isi":   -2.73,
    " some": -2.98,
    ",":     -3.03,
    "1":     -3.25,
    " what": -3.43
}
```

Logit bias lets the user “boost” the probability of specific tokens to put them in the top-k:

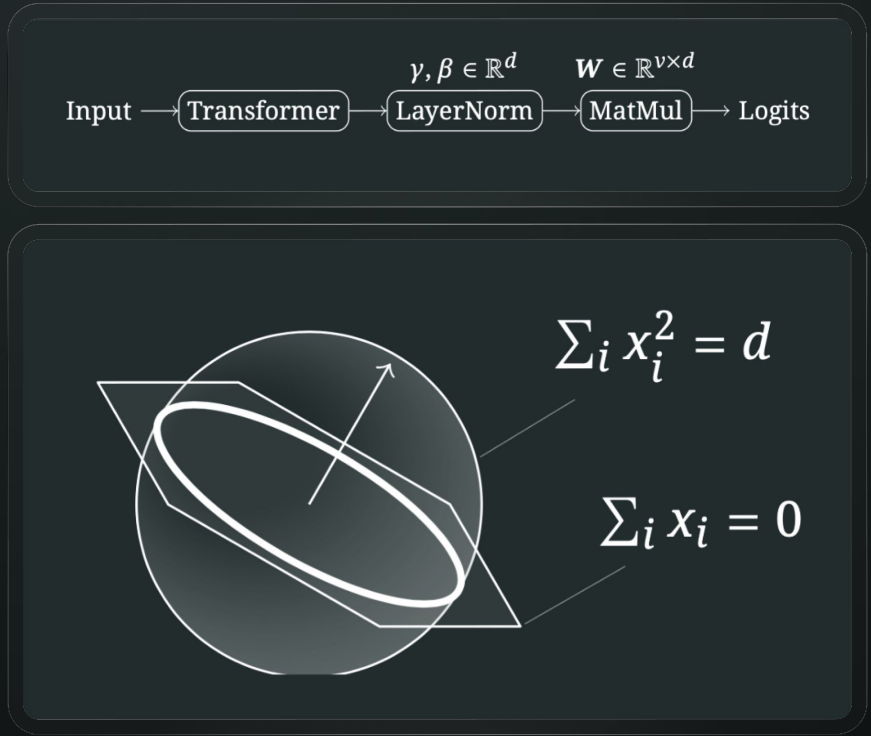
```
LOGIT BIAS
{“ hi”: +100}

BIASED TOP-K OUTPUT
{
    “ hi”: -1.02,
    “isi”: -3.23,
     …
}
```

https://help.openai.com/en/articles/5247780-using-logit-bias-to-alter-token-probability-with-the-openai-api

Why Logits Leak Info: The Softmax Bottleneck

- Because of that fixed size, the outputs can't cover the entire vocabulary patterns of the LLM (GPT-3.5 has a vocabulary of >50,000 tokens)

- They're confined to this smaller d-dimensional subspace
  → the **softmax bottleneck**



Exploiting the Bottleneck

- Attackers send many diverse prompts to the API
- Collect the output probabilities/logits for each token
- Use SVD to find how many unique “directions” the outputs span
- The number of directions ≈ the hidden size of the model

Cost estimation on GPT-3.5-Turbo API

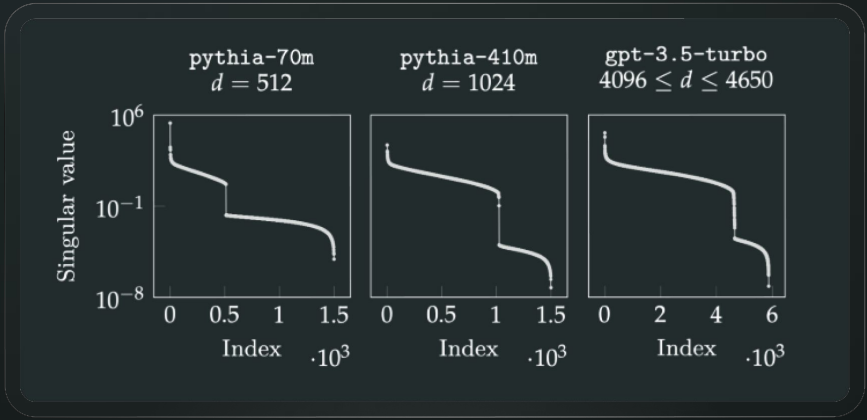
Algorithm	Complexity	API calls per output	Image price (USD)
Logprob-free (Morris et al., 2023)	$v \log(\beta_{\max}/\epsilon)$	800 000	16 384
With logprobs	v/k	20 000	410

Matthew et al. (2024), Morris et al. (2023), Carlini et al. (2024)

Case Study:GPT3.5-turbo

- ~6,000 API queries
- Analysis showed outputs stopped adding new info after ~4,100 dimensions
- Hidden size inferred: ≈4,096
- Enabled estimates of GPT3.5-Turbo's true scale without access to its weights

(Matthew et al., 2024)



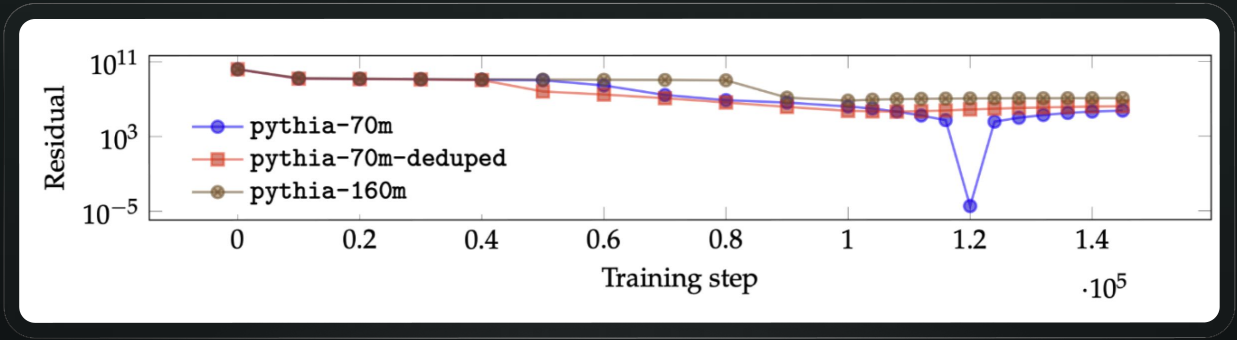
Applying the inverse log-softmax to the logprobs we get the logits L. All that remains is to check the rank of the logits (L)!

API outputs can reveal a model's true scale, letting outsiders estimate cost, capability, and even catch providers swapping or lying about their models.

LLM images: threat or benefit?

APIs leak more than just the hidden size, they leak the column space of W. This can be useful for

- Identifying which model generated an output



(Matthew et al., 2024)

LLM images: threat or benefit?

APIs leak more than just the hidden size, they leak the column space of W. This can be useful for

- Auditing changes in an API LLM

Change	Interpretation
No logit change, no image change	No update
Logit change, no image change	Hidden prompt change or partial finetune
Low-rank image change (§7.1)	LoRA update
Image change	Full finetune

Table 2: Implications of image/logit changes.

(Matthew et al., 2024)

Security Risks from Softmax Leakage

Risk 1

Enables model stealing via distillation or extraction attacks

Risk 2

Lets competitors infer model scale and architecture

Risk 3

Opens the door to crafting adversarial attacks using leaked behavior

Risk 4

Low-cost feasibility as attacks can succeed for hundreds to a few thousand USD via API access

Opportunities: Turning Leakage into Accountability

Independent Audits

verify a provider's model size, safety claims, or fine-tune dates

Change Detection

spot silent model swaps or unannounced updates in production

Provenance Checks

confirm an API isn't a re-branded open-source model

Transparency Signals

leakage can act as a public "fingerprint," building user trust

Policy & Compliance

regulators can verify that licensed or safety-critical models stay unchanged

Key Takeaways

- What do LLM APIs reveal about the model?
- What useful features can APIs keep without revealing model information?
- How can we verify the source of LLM outputs?

xiangren@usc.edu | 𝕏 @xiangrennlp

Logits of API-Protected LLMs Leak Proprietary Information

Matthew Finlayson **Xiang Ren** **Swabha Swayamdipta**
Thomas Lord Department of Computer Science
University of Southern California
{mfinlays, xiangren, swabhas}@usc.edu

Abstract

Large language model (LLM) providers often hide the architectural details and parameters of their proprietary models by restricting public access to a limited API. In this work we show that, with only a conservative assumption about the model architecture, it is possible to learn a surprisingly large amount of non-public information about an API-protected LLM from a relatively small number of API queries (e.g., costing under $1000 USD for OpenAI's `gpt-3.5-turbo`). Our findings are centered on one key observation: most modern LLMs suffer from a softmax bottleneck, which restricts the model outputs to a linear subspace of the full output space. We exploit this fact to unlock several capabilities, including (but not limited to) obtaining cheap full-vocabulary outputs, auditing for specific types of model updates, identifying the source LLM given a single full LLM output, and even efficiently discovering the LLM's hidden size. Our empirical investigations show the effectiveness of our methods, which allow us to estimate the embedding size of OpenAI's `gpt-3.5-turbo` to be about 4096. Lastly, we discuss ways that LLM providers can guard against these attacks, as well as how these capabilities can be viewed as a feature (rather than a bug) by allowing for greater transparency and accountability.

https://mattf1n.github.io/

Future

How can we trust LLM providers?

How can we verify the source of LLM outputs?

How can we trace the contributions in AI development?

How to achieve fair compensation in a new AI economy?

Sahara AI

The Convergence of AI and Web3

Exploring the Infra Needed for a Collaborative, User-Owned AI Economy

SaharaAI.com @SaharaLabsAI #AIForAll

Sahara AI

AI Is Already One of the Most Transformative Technologies Since the Internet...

- AI: **$15.7T** projected impact on global GDP by 2030
- **78%** of enterprises now report using AI in at least one core function
- ChatGPT has **~800 million** weekly active users

Sahara AI

The Convergence of AI x Web3

- AI introduces **new digital primitives**: agents, datasets, and models that hold real-world value
- Web3 offers the tools for transparent ownership, on-chain attribution, and trustless collaboration

Sahara AI

The Convergence of AI x Web3

- AI introduces **new digital primitives:** agents, datasets, and models that hold real-world value
- Web3 offers the tools for transparent ownership, composability, and trustless collaboration
- Together, they unlock the possibility of building autonomous systems that are not only intelligent but governed and monetized by their communities

Sahara AI

But AI is missing the critical infra layer needed to make this happen

Sahara AI

Building the Foundations for a Collaborative AI Economy

- To realize the promise of AI x Web3, we need more than theoretical alignment, we need real, usable infrastructure
- That means protocols for attribution, platforms for development, and systems that can scale across chains

Sahara AI

Building the Foundations for a Collaborative AI Economy

- To realize the promise of AI x Web3, we need more than theoretical alignment, we need real, usable infrastructure
- That means protocols for attribution, platforms for development, and systems that can scale across chains
- Sahara AI was designed to meet that need: not as a single app, but as an ecosystem foundation where open participation, verifiable contribution, and fair monetization are built in

Sahara AI

The Problem we are solving

Imagine you spent months curating a high-quality dataset or training an AI model. Today, once you share that dataset or model, you lose control over how it's used and receive no compensation when someone builds a billion-dollar AI application on top of your work.

Sahara AI

Economic Rails for Decentralized AI

Three Main Benefits:

Sahara AI

Economic Rails for Decentralized AI

Three Main Benefits:

01.

Fair Revenue:

Every contributor gets paid automatically based on their impact

Sahara AI

Economic Rails for Decentralized AI

Three Main Benefits:

01.

Fair Revenue:

Every contributor gets paid automatically based on their impact

02.

Verifiable Attribution:

Immutable on-chain proof of who built what

Sahara AI

Economic Rails for Decentralized AI

Three Main Benefits:

01.

Fair Revenue:

Every contributor gets paid automatically based on their impact

02.

Verifiable Attribution:

Immutable on-chain proof of who built what

03.

Composable Licensing:

Programmable usage rights that travel with the asset

Sahara AI

Why This Matters

Persona	Pain Points	Solution
Enterprises & Data Providers	Lose control and royalties once data is shared	Tokenize → License → Earn residuals recursively
AI Developers & Fine-tuners	Hard to monetize open-source contributions	Contribution NFTs + Staking Pool yield
Web3 Builders	No AI-native economic primitives	Plug-and-play NFTs, Vaults, Cashiers

Sahara AI

Real-World Flows:
Registering and Monetizing a Dataset

Sahara AI

Real-World Flows:
Registering and Monetizing a Dataset

01.

Registration:

Sarah uploads metadata and receives an AssetOwnerNFT with a Token-Bound Account.

Sahara AI

Real-World Flows:
Registering and Monetizing a Dataset

01.

Registration:

Sarah uploads metadata and receives an AssetOwnerNFT with a Token-Bound Account.

02.

Setting Terms:

Sarah sets $100 research and $10K commercial licenses via TBA. Royalties: 60% hers, 20% team, 20% hospital.

Sahara AI

Real-World Flows:
Registering and Monetizing a Dataset

01. 

Registration:

Sarah uploads metadata and receives an AssetOwnerNFT with a Token-Bound Account.

02. 

Setting Terms:

Sarah sets $100 research and $10K commercial licenses via TBA. Royalties: 60% hers, 20% team, 20% hospital.

03. 

First Sale:

A pharma company buys a commercial license. The $10K flows through the Cashier to the dataset's Vault.

Sahara AI

Real-World Flows:
Registering and Monetizing a Dataset

01.

Registration:

Sarah uploads metadata and receives an AssetOwnerNFT with a Token-Bound Account.

02.

Setting Terms:

Sarah sets $100 research and $10K commercial licenses via TBA. Royalties: 60% hers, 20% team, 20% hospital.

03.

First Sale:

A pharma company buys a commercial license. The $10K flows through the Cashier to the dataset's Vault.

04.

Automatic Distribution:

The Vault splits: $6K to Sarah, $2K to contributors, $2K to the hospital.

Sahara AI

Real-World Flows:
Registering and Monetizing a Dataset

01.

Registration:

Sarah uploads metadata and receives an AssetOwnerNFT with a Token-Bound Account.

02.

Setting Terms:

Sarah sets $100 research and $10K commercial licenses via TBA. Royalties: 60% hers, 20% team, 20% hospital.

03.

First Sale:

A pharma company buys a commercial license. The $10K flows through the Cashier to the dataset's Vault.

04.

Automatic Distribution:

The Vault splits: $6K to Sarah, $2K to contributors, $2K to the hospital.

05.

Downstream Revenue:

A model trained on Sarah's data earns revenue. She automatically gets 10% of its earnings, forever.

Sahara AI

Real-World Flows:
Collaborative Model Development

Sahara AI

Real-World Flows:
Collaborative Model Development

01.

Base Model Registration:

Team lead Alex registers the base model, receiving the AssetOwnerNFT.

Sahara AI

Real-World Flows:
Collaborative Model Development

01.

Base Model Registration:

Team lead Alex registers the base model, receiving the AssetOwnerNFT.

02.

Contributions Flow In:

- Bob fine-tunes the model for medical applications
- Carol adds safety filters
- Diana adds data, each minted as a validated ContributionNFT.

Sahara AI

Real-World Flows:
Collaborative Model Development

01.

Base Model Registration:

Team lead Alex registers the base model, receiving the AssetOwnerNFT.

02.

Contributions Flow In:

- Bob fine-tunes the model for medical applications
- Carol adds safety filters
- Diana adds data, each minted as a validated ContributionNFT.

03.

Revenue Sharing Setup:

The team sets splits via the Vault. Each ContributionNFT holder earns a share by contribution.

Sahara AI

Real-World Flows:
Collaborative Model Development

01.

Base Model Registration:

Team lead Alex registers the base model, receiving the AssetOwnerNFT.

02.

Contributions Flow In:

- Bob fine-tunes the model for medical applications
- Carol adds safety filters
- Diana adds data, each minted as a validated ContributionNFT.

03.

Revenue Sharing Setup:

The team sets splits via the Vault. Each ContributionNFT holder earns a share by contribution.

04.

Ongoing Rewards:

As the model is used commercially, revenue automatically flows to all contributors according to their stakes.

Sahara AI

4 Key Asset Ownership Questions

Sahara Protocol seeks to provide **on-chain answers** to four foundational asset questions:

Module Name	Definition
Who owns the asset?	Clear sovereignty and governance rights
Who contributed?	Contributor structure for attribution and future revenue participation
Who can use it?	On-chain usage rights and authorization proof
Is it legitimate?	On-chain verifiability rather than off-chain negotiation

Sahara AI

Token-Bound Accounts (TBAs):

Giving Assets Their Own Wallets

Sahara AI

Token-Bound Accounts (TBAs):

Giving Assets Their Own Wallets

01.

The asset can receive payments directly

02.

It can sign contracts and issue licenses

03.

It can pay out royalties to contributors

04.

When you transfer ownership of the NFT, you transfer control of everything the asset owns

Sahara AI

The License System: Programmable Usage Rights

Commercial vs. non-commercial use

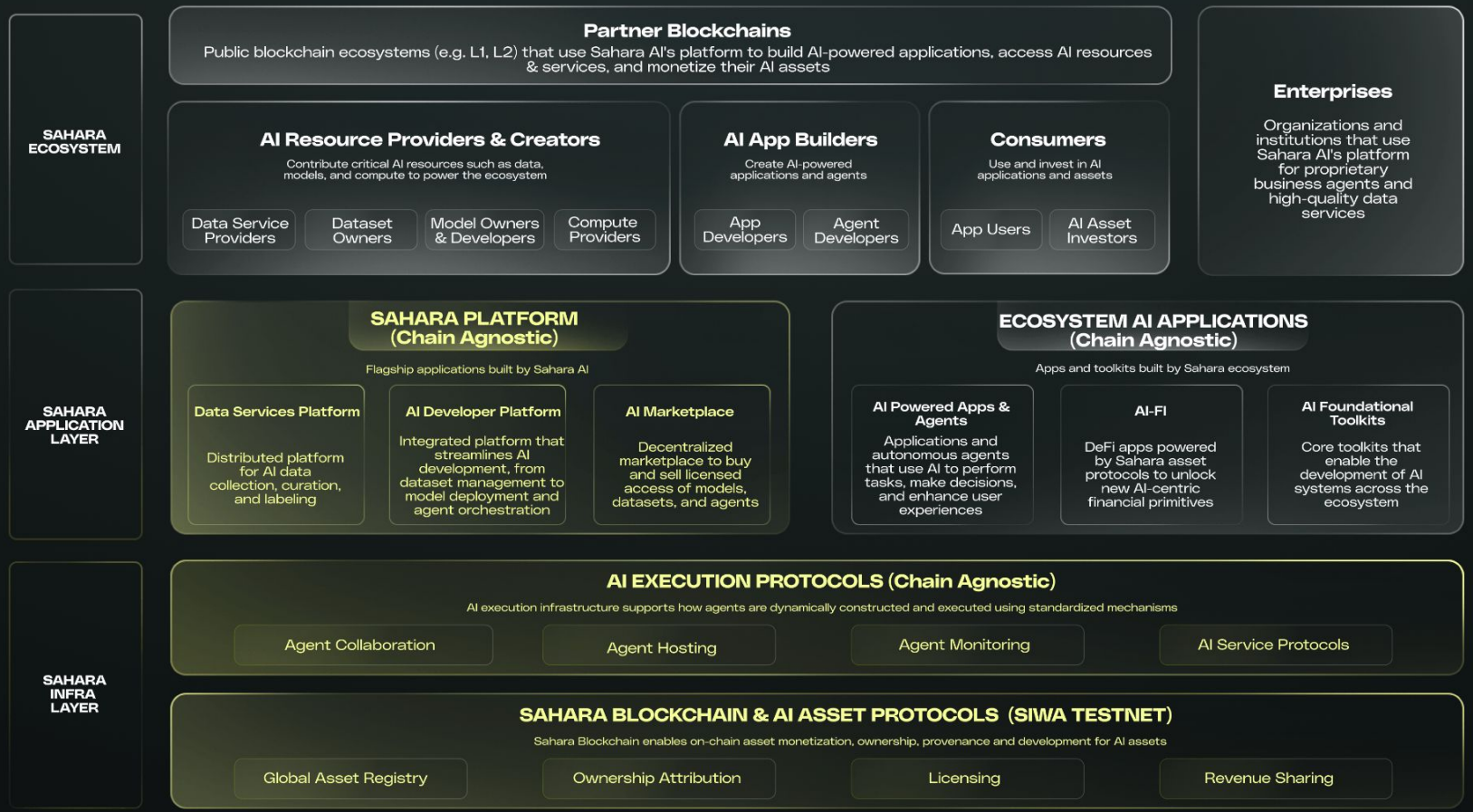
Whether derivatives are allowed

Time limitations

Geographic restrictions

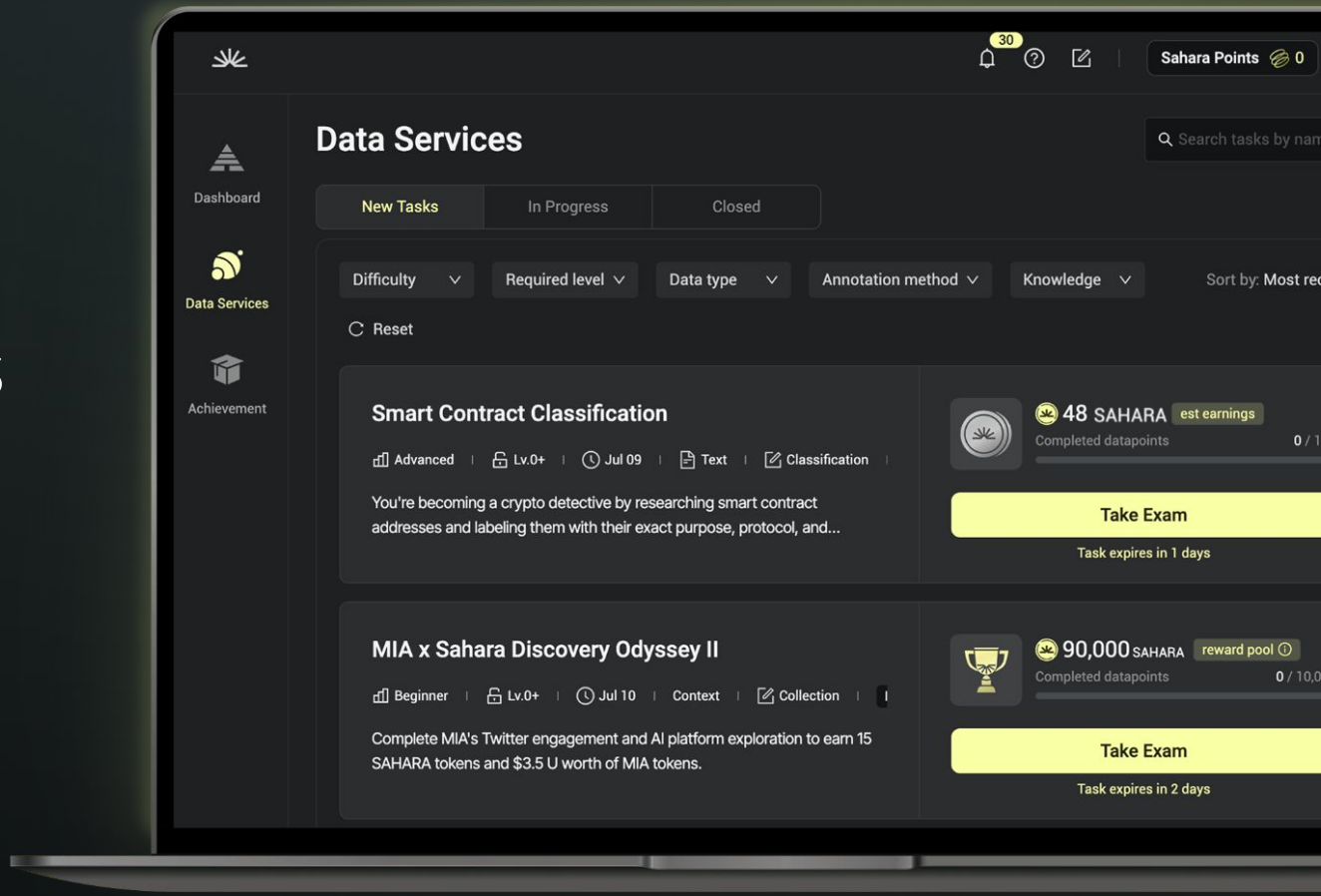
Custom terms

Sahara AI


SAHARA ECOSYSTEM
Partner Blockchains
Public blockchain ecosystems (e.g. L1, L2) that use Sahara AI's platform to build AI-powered applications, access AI resources & services, and monetize their AI assets
AI Resource Providers & Creators
Contribute critical AI resources such as data, models, and compute to power the ecosystem
Data Service Providers
Dataset Owners
Model Owners & Developers
Compute Providers
AI App Builders
Create AI-powered applications and agents
App Developers
Agent Developers
Consumers
Use and invest in AI applications and assets
App Users
AI Asset Investors
Enterprises
Organizations and institutions that use Sahara AI's platform for proprietary business agents and high-quality data services
SAHARA APPLICATION LAYER
SAHARA PLATFORM (Chain Agnostic)
Flagship applications built by Sahara AI
Data Services Platform
Distributed platform for AI data collection, curation, and labeling
AI Developer Platform
Integrated platform that streamlines AI development, from dataset management to model deployment and agent orchestration
AI Marketplace
Decentralized marketplace to buy and sell licensed access of models, datasets, and agents
ECOSYSTEM AI APPLICATIONS (Chain Agnostic)
Apps and toolkits built by Sahara ecosystem
AI Powered Apps & Agents
Applications and autonomous agents that use AI to perform tasks, make decisions, and enhance user experiences
AI-FI
DeFi apps powered by Sahara asset protocols to unlock new AI-centric financial primitives
AI Foundational Toolkits
Core toolkits that enable the development of AI systems across the ecosystem
SAHARA INFRA LAYER
AI EXECUTION PROTOCOLS (Chain Agnostic)
AI execution infrastructure supports how agents are dynamically constructed and executed using standardized mechanisms
Agent Collaboration
Agent Hosting
Agent Monitoring
AI Service Protocols
SAHARA BLOCKCHAIN & AI ASSET PROTOCOLS (SIWA TESTNET)
Sahara Blockchain enables on-chain asset monetization, ownership, provenance and development for AI assets
Global Asset Registry
Ownership Attribution
Licensing
Revenue Sharing

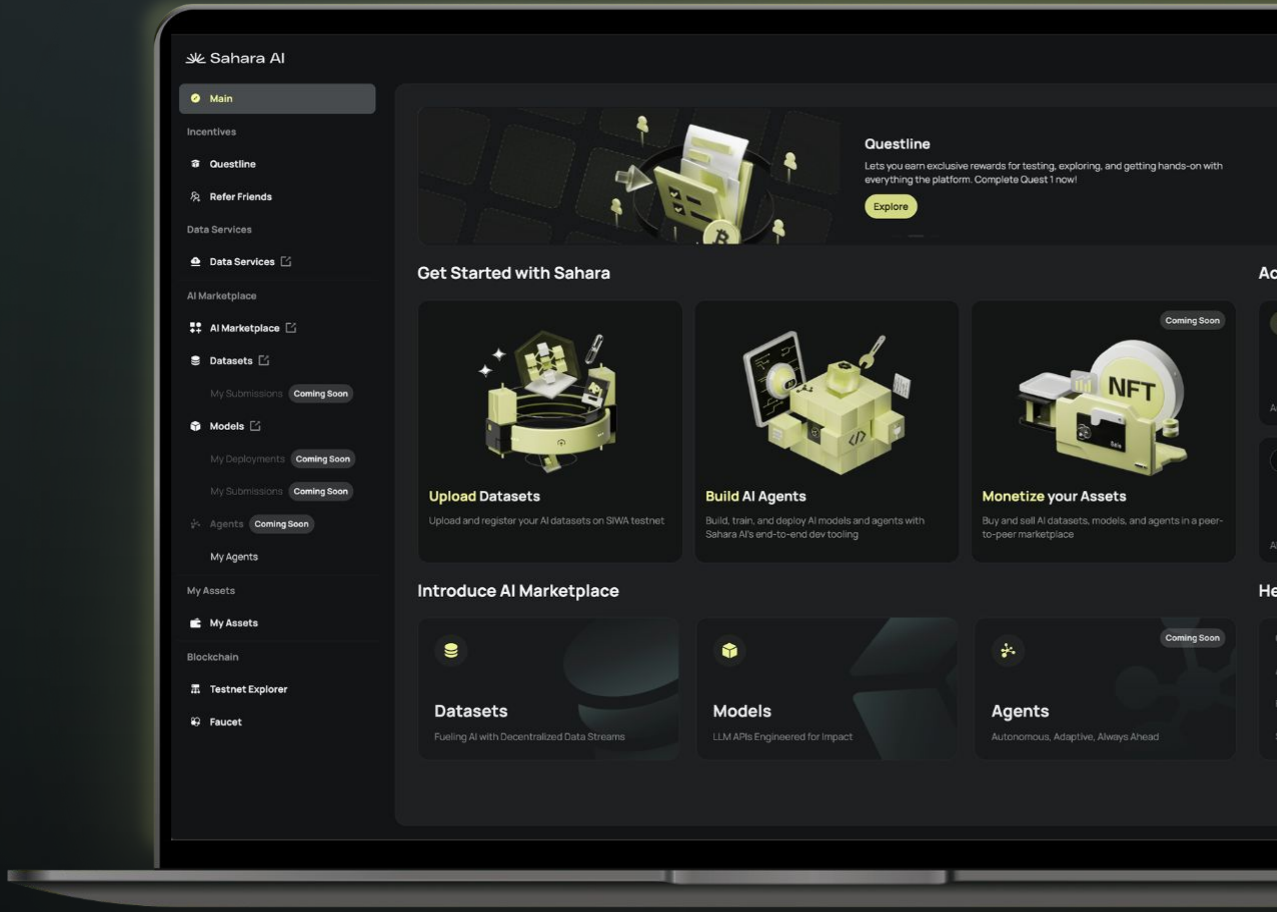
Sahara AI

Data Services Platform



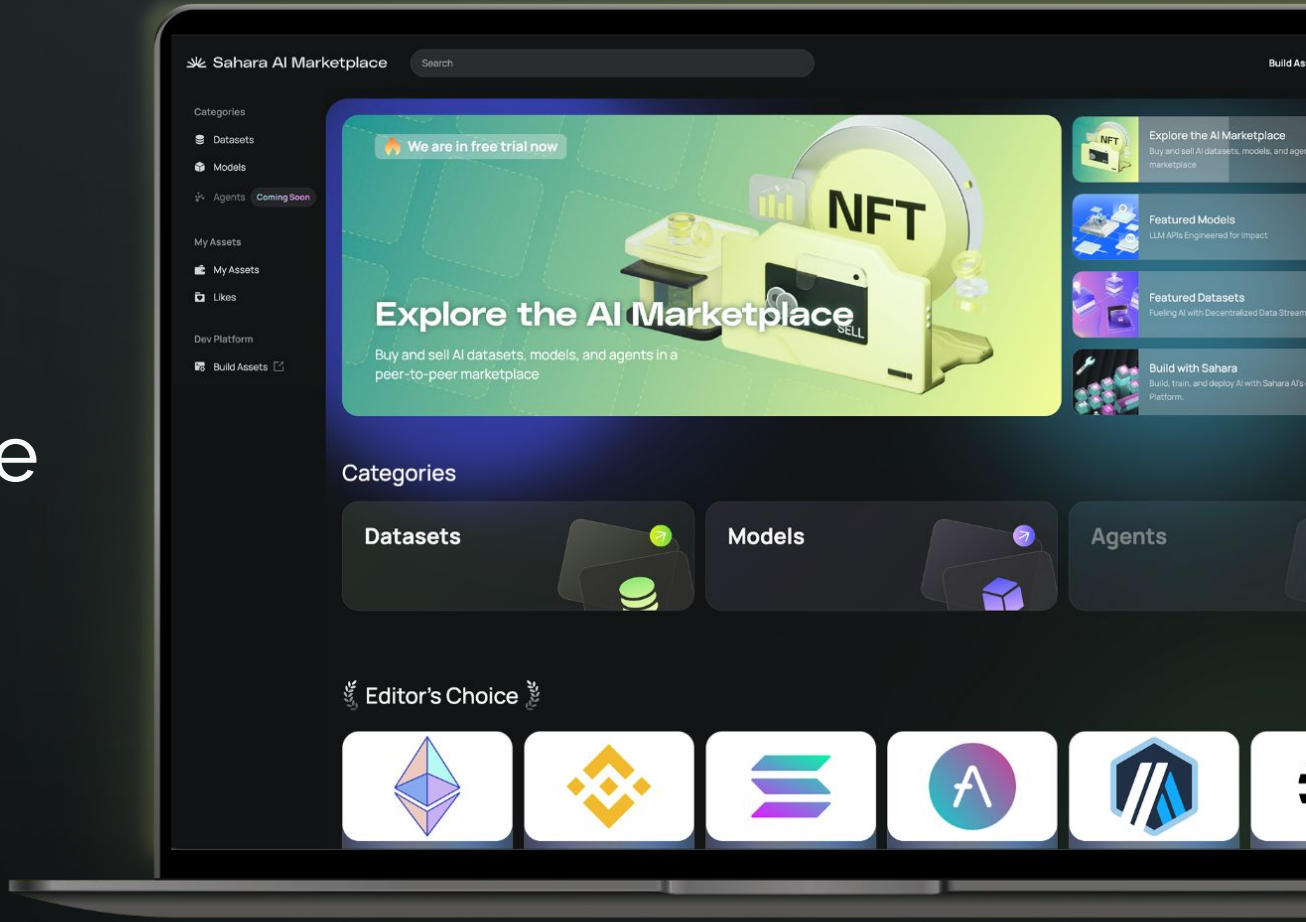
Sahara AI

AI Developer Platform



Sahara AI

AI Marketplace



Sahara AI

We must rewrite the relationship between AI companies, users, and contributors, to create a more equitable and transparent ecosystem for all.

Sahara AI

An AI-Driven Future Is Inevitable.

Sahara AI

An AI-Driven Future Is Inevitable.

An Open and Equitable One Isn't.

Sahara AI

Join the Collaborative AI Economy

SaharaAI.com @SaharaLabsAI #AIForAll