Clarity: A Crowdsourced Classification Framework for AI-Generated Content Detection on YouTube

Spenser Wu Caroline Zhu

Abstract

This paper presents Clarity, an open-source Chrome browser extension that leverages crowdsourced labeling and machine learning to identify and filter AI-generated content on YouTube. We formalize a consensus mechanism that aggregates user classifications while providing probabilistic guarantees against adversarial manipulation. The framework employs lightweight pseudonymous identity, majority voting with accuracy tracking, and a shared blacklist architecture. We derive conditions under which honest consensus emerges and characterize the system's robustness to coordinated attacks. The current implementation integrates an XGBoost classification model achieving 72% accuracy (0.767 ROC-AUC) using 21 engineered features extracted from YouTube channel metadata. This paper documents both the theoretical foundations and practical implementation details, including feature engineering, model selection, and the server-side inference pipeline. We offer this framework as a starting point for the broader community and invite collaboration from researchers and engineers interested in solving the AI content authenticity problem at scale.

1 Introduction

The proliferation of AI-generated video content on YouTube presents a highly urgent content moderation challenge. Unlike traditional spam or policy violations, AI-generated content exists on a spectrum of authenticity and may not violate platform terms of service. Users seeking to curate their viewing experience currently lack tools to systematically identify and filter such content.

Existing crowdsourced moderation systems demonstrate the viability of distributed content labeling. SponsorBlock, a browser extension with over one million active users, allows viewers to skip sponsored segments in YouTube videos by crowdsourcing timestamp labels from the community. With sufficient participation, only a small fraction of users need to label content for the entire user base to benefit, a powerful network effect that transforms individual effort into collective value.

However, AI content detection differs from sponsor segment identification in three critical dimensions:

(1) **Scale:** AI-generated content volume vastly exceeds sponsored segments. While a video has at most a few sponsor reads, YouTube hosts millions of channels with varying degrees of AI involvement.

(2) **Ambiguity:** Sponsor segments have clear boundaries; AI content detection requires subjective judgment about authenticity, intent, and degree.

(3) **Adversarial dynamics:** Sponsors have no incentive to evade SponsorBlock. AI content creators, by contrast, may actively resist classification.

We propose Clarity, a framework that addresses these challenges through a hybrid approach: a machine learning classifier provides automated predictions for unreviewed channels, while a crowdsourced consensus mechanism operates at the channel level to aggregate human judgments, dramatically reducing the labeling burden while capturing the observation that AI-generated content tends to cluster by creator.

2 Problem Formalization

2.1 Definitions

Let $\mathcal{C}$ denote the set of all YouTube channels and $\mathcal{U}$ the set of registered users. Each channel $c \in \mathcal{C}$ has a true label $y(c) \in \{0, 1\}$, where $y(c) = 1$ indicates the channel primarily produces AI-generated content. This ground truth is unobservable but fixed.

Definition 1 (AI-Generated Content)**.** *For purposes of this framework, we define a channel as "AI-generated" if the majority of its recent content exhibits one or more of the following characteristics: (1) synthetic voiceover (text-to-speech narration), (2) programmatically generated or heavily templated visuals (stock footage compilations, AI-generated imagery, slideshow formats), or (3) minimal human editorial judgment in content selection and presentation. This is an operational definition that admits edge cases—channels using AI tools for editing, translation, or accessibility assistance are not considered AI-generated unless the core creative content itself is synthetic. We acknowledge this boundary is ultimately a judgment call, but one that human labelers can apply with reasonable consistency.*

Each user $u \in \mathcal{U}$ may submit a vote $v(u, c) \in \{0, 1\}$ for channel c. We model user reliability as a single parameter:

Definition 2 (Accuracy)**.** *The accuracy of user u is $\alpha(u) \coloneqq P(v(u, c) = y(c))$, the probability that user u votes correctly, assumed constant across channels.*

For the MVP, we assume users are either honest ($\alpha > 0.5$) or adversarial ($\alpha < 0.5$), with no strategic behavior beyond consistent voting patterns.

2.2 Threat Model

We consider an adversary who controls a fraction f of registered accounts and seeks to manipulate channel classifications. The adversary's capabilities are constrained by:

(1) **Account creation cost:** Each pseudonymous identity requires a persistent browser instance (or programmatic management of local storage), imposing marginal cost κ per Sybil account.

(2) **Rate limiting:** Each account may submit at most r votes per time period, preventing automated scripts from overwhelming the vote pool.

(3) **Detection risk:** Accounts voting against eventual consensus accumulate negative track records and face rate limiting or shadowbanning.

The adversary's objective is to flip the classification of a target channel from $\hat{y}(c) = 1$ (correctly identified as AI) to $\hat{y}(c) = 0$ (incorrectly cleared), or vice versa.

Scope limitations. This threat model assumes unsophisticated adversaries who vote consistently against ground truth. A more sophisticated attacker could vote honestly on most channels to build

a high accuracy score, then deploy that credibility on high-value targets. Such *strategic adversaries* would evade our accuracy-tracking mechanism. We consider this attack class out of scope for the MVP; the trust-weighted voting extension (Section 7) and Bayesian aggregation methods provide partial mitigation by modeling per-channel reliability rather than global accuracy.

We explicitly accept weaker identity guarantees in exchange for reduced user friction. The security model relies on rate limiting and consensus-based detection rather than identity verification. Section 7 discusses stronger identity mechanisms for future versions.

3 Machine Learning Classification

While crowdsourced voting provides ground truth labels, it cannot scale to the millions of YouTube channels that users may encounter. To address the cold-start problem for unlabeled channels, Clarity integrates a machine learning classifier that predicts AI-generated content based on observable channel metadata.

3.1 Model Architecture

The production classifier uses XGBoost (Extreme Gradient Boosting), a tree ensemble method selected through systematic comparison against logistic regression baseline.

Model	**Accuracy**	**ROC-AUC**	**F1 Score**
Logistic Regression (Baseline)	68.56%	0.729	0.623
XGBoost (Production)	72.02%	0.767	0.651
Improvement	+3.46 pp	+0.038	+0.028

Table 1: 5-fold cross-validation performance comparison (n=3,217 labeled samples). Improvements shown as absolute differences (percentage points for accuracy, raw values for AUC/F1).

XGBoost was selected for production due to its significantly higher ROC-AUC and precision (70.42% vs 64.87%), allowing for more reliable initial AI detection with fewer false positives. The model hyperparameters are:

- `n_estimators`: 100 (number of boosting rounds)
- `max_depth`: 5 (tree depth limit)
- `learning_rate`: 0.1 (shrinkage parameter)

3.2 Feature Engineering

The classifier operates on 21 engineered features extracted from YouTube channel metadata via the YouTube Data API v3. Features are organized into three categories based on their signal type and derivation.

3.2.1 Primary Volume Metrics

Six features capture raw channel statistics, log-transformed ($\log(1 + x)$) to reduce skew:

$$\texttt{sub\_count} : \text{Total channel subscribers}$$
$$\texttt{video\_count} : \text{Total number of uploaded videos}$$
$$\texttt{view\_count} : \text{Aggregate channel views}$$
$$\texttt{avg\_view\_count\_recent} : \text{Mean views across most recent 50 videos}$$
$$\texttt{avg\_like\_count\_recent} : \text{Mean likes across most recent 50 videos}$$
$$\texttt{avg\_comment\_count\_recent} : \text{Mean comments across most recent 50 videos}$$

3.2.2 Derived Engagement & Efficiency Ratios

Eight features capture relationships between metrics that may distinguish AI-generated content from organic channels:

$$\texttt{engagement\_rate} = \frac{\text{Likes}_{\text{recent}} + \text{Comments}_{\text{recent}}}{\text{Views}_{\text{recent}}}$$
$$\texttt{views\_to\_comments} = \frac{\texttt{avg\_view\_count\_recent}}{\texttt{avg\_comment\_count\_recent} + 1}$$
$$\texttt{views\_to\_likes} = \frac{\texttt{avg\_view\_count\_recent}}{\texttt{avg\_like\_count\_recent} + 1}$$
$$\texttt{comments\_to\_likes} = \frac{\texttt{avg\_comment\_count\_recent}}{\texttt{avg\_like\_count\_recent} + 1}$$
$$\texttt{videos\_per\_day} = \frac{\texttt{video\_count}}{\texttt{channel\_age\_days} + 1}$$
$$\texttt{views\_per\_video} = \frac{\texttt{view\_count}}{\texttt{video\_count} + 1}$$
$$\texttt{subs\_per\_video} = \frac{\texttt{sub\_count}}{\texttt{video\_count} + 1}$$
$$\texttt{views\_per\_sub} = \frac{\texttt{view\_count}}{\texttt{sub\_count} + 1}$$

The addition of 1 in denominators prevents division by zero while maintaining interpretability.

3.2.3 Content Metadata

Seven features capture channel metadata and content characteristics:

$$\texttt{description\_len} : \text{Character count of channel description}$$
$$\texttt{topic\_count} : \text{Number of YouTube topic categories}$$
$$\texttt{channel\_age\_days} : \text{Age of channel in days since creation}$$
$$\texttt{uploads\_per\_month} : \text{Video posting frequency}$$
$$\texttt{hd\_ratio} : \text{Fraction of recent videos in HD}$$
$$\texttt{caption\_ratio} : \text{Fraction of recent videos with captions}$$
$$\texttt{avg\_video\_duration} : \text{Mean video length in seconds}$$

3.3 Training Pipeline

The model training pipeline consists of three stages:

Stage 1: Data Collection. Training data is collected from manually labeled channels stored in a Supabase database. The current dataset contains 3,217 labeled samples with binary labels (`is_ai` $\in \{0, 1\}$).

Stage 2: Feature Preprocessing.

1. Derived features are computed from base channel statistics (Section 3.2.2)
2. Log transform ($\log(1 + x)$) is applied to volume metrics to reduce skewness
3. Missing values are filled with 0
4. StandardScaler normalizes features to zero mean and unit variance

Stage 3: Model Training. The XGBoost classifier is trained on the preprocessed features. Model selection uses 5-fold cross-validation, with ROC-AUC as the primary metric due to its robustness to class imbalance.

The trained model is serialized as a scikit-learn Pipeline containing both the StandardScaler and XGBClassifier, ensuring consistent preprocessing during inference.

3.4 Inference Architecture

Channel classification follows a server-side inference pattern to protect API credentials:

1. **Request:** Chrome extension sends channel handle to backend API
2. **Cache Check:** Server queries Supabase for cached analysis (TTL: 7 days)
3. **Feature Extraction:** On cache miss, server calls YouTube Data API v3:
   - `channels.list` — Channel metadata and statistics
   - `playlistItems.list` — Recent video IDs (max 50)
   - `videos.list` — Video statistics and content details
4. **Feature Engineering:** Derived features computed, log transforms applied
5. **Prediction:** XGBoost model returns $(\hat{y}, p)$ where $\hat{y} \in \{0, 1\}$ and $p \in [0, 1]$ is the AI probability
6. **Caching:** Results stored in Supabase for future requests

This architecture requires only 3 YouTube API calls per unique channel, with subsequent requests served from cache.

4 Consensus Mechanism

4.1 Vote Aggregation

For MVP simplicity, we employ unweighted majority voting. Let $V(c) = \{u : u \text{ voted on } c\}$ denote the set of users who voted on channel c. The estimated label is:

$$\hat{y}(c) = \begin{cases} 1 & \text{if } \sum_{u \in V(c)} v(u,c) > \dfrac{|V(c)|}{2} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

A channel enters the shared blacklist when $\hat{y}(c) = 1$ and $|V(c)| \geq n_{\min}$, where $n_{\min}$ is the minimum vote threshold.

In plain terms: a channel is blacklisted if (1) a majority of voters labeled it AI-generated, and (2) at least $n_{\min}$ total votes were cast.

ML classifier integration. When a channel has not received sufficient votes ($|V(c)| < n_{\min}$), the ML classifier's prediction serves as a prior estimate, displayed to users with a confidence score. Importantly, the classifier's prediction does *not* influence which channels are surfaced for voting—users vote on channels they encounter organically through YouTube's interface. This design prevents feedback loops where classifier biases could be amplified through selective labeling. The classifier and consensus mechanism operate independently: the classifier handles cold-start predictions, while human votes provide ground truth that supersedes ML predictions once $n_{\min}$ is reached.

4.2 Accuracy Tracking

Each user maintains a track record based on agreement with consensus outcomes. Define:

$$a(u) \coloneqq |\{c : v(u,c) = \hat{y}(c)\}| \qquad \text{(agreement count)} \tag{2}$$

$$n(u) \coloneqq |\{c : u \text{ voted on } c\}| \qquad \text{(total votes)} \tag{3}$$

$$\hat{\alpha}(u) \coloneqq \frac{a(u)}{n(u)} \qquad \text{(observed accuracy)} \tag{4}$$

The core assumption is that honest users agree with eventual consensus more often than adversarial ones. Systematic disagreement therefore signals either poor judgment or malicious intent—both warrant reduced influence.

Users with $\hat{\alpha}(u) < \alpha_{\min}$ after $n(u) \geq n_{\text{calibration}}$ votes may be flagged for review or rate-limited. For MVP, we recommend $\alpha_{\min} = 0.3$ and $n_{\text{calibration}} = 20$.

Circularity limitation. We acknowledge that this accuracy-tracking mechanism has a circularity problem: user accuracy is defined as agreement with consensus, but consensus is determined by user votes. This means early voters on a channel disproportionately influence what "correct" means for that channel. With $n_{\min} = 5$, the first 3 agreeing voters essentially define ground truth. This bootstrapping issue is inherent to crowdsourced labeling systems without external ground truth. The Dawid-Skene extension described in Section 7 partially addresses this by jointly inferring user reliabilities and true labels, breaking the circular dependency through probabilistic modeling.

4.3 Theoretical Guarantees

Under the assumption that honest users have accuracy $\alpha_h > 0.5$ and constitute fraction $(1 - f)$ of voters, we derive finite-sample bounds on consensus error.

Proposition 1 (Consensus Error Bound)**.** *Consider a channel c with n total voters, of whom fraction $(1 - f)$ are honest with accuracy $\alpha_h > 0.5$, and fraction $f < 0.5$ are adversarial (voting incorrectly*

with probability 1). Let $\mu = \alpha_h(1-f) + (1-f)(1-\alpha_h) \cdot 0 + f \cdot 0 = \alpha_h(1-f)$ *for an AI channel. The probability of incorrect consensus is bounded by:*

$$P(\textit{incorrect consensus}) \leq \exp\left(-2n\left(\alpha_h(1-f) - 0.5\right)^2\right)$$

provided $\alpha_h(1-f) > 0.5$.

Proof sketch. Each vote is a Bernoulli random variable. For an AI channel, honest users vote correctly with probability α_h, and adversarial users vote incorrectly with probability 1. The expected fraction of correct votes is $\alpha_h(1-f)$. By Hoeffding's inequality, the probability that the sample mean falls below 0.5 (causing incorrect consensus) decays exponentially in n. □

Numerical example. With $\alpha_h = 0.7$ (honest users correct 70% of the time) and $f = 0.2$ (20% adversarial accounts), we have $\alpha_h(1-f) = 0.56$. For $n = 20$ voters, the error probability is at most $\exp(-2 \cdot 20 \cdot 0.06^2) \approx 0.86$—not yet reliable. At $n = 100$, this drops to ≈ 0.49. At $n = 500$, error probability falls below 0.01. This illustrates that meaningful guarantees require substantial community participation.

Proposition 2 (Attack Cost Lower Bound)**.** *To flip a classification with* n *honest votes, an adversary must control at least* $n+1$ *accounts, incurring minimum cost* $(n+1)\kappa$.

In plain terms: if decisions are made by majority vote, an attacker needs more fake voters than real voters to change the outcome. As community participation grows, this becomes increasingly difficult to achieve.

5 System Architecture

5.1 Identity and Sybil Resistance

Each browser installation generates a persistent pseudonymous identifier stored in local browser storage. This provides:

(1) **Lightweight onboarding:** No account creation required; users can vote immediately upon installing the extension.

(2) **Basic Sybil resistance:** Attackers must maintain persistent browser instances (or programmatically manage local storage), imposing non-zero marginal cost.

(3) **Behavioral detection:** Accounts systematically disagreeing with consensus are rate-limited or shadowbanned per Section 4.2.

We explicitly trade stronger identity guarantees for reduced friction, following the precedent set by SponsorBlock. The security model relies on rate limiting and consensus-based detection rather than identity verification. Future versions may introduce optional verified accounts (e.g., via OAuth) with higher vote weights for users who choose to authenticate.

5.2 Data Model

The system maintains three primary data structures:

$$\begin{aligned}\texttt{Channel} &: \texttt{channel\_id} \to (\texttt{vote\_count\_ai}, \texttt{vote\_count\_human}, \texttt{status}) \\ \texttt{Vote} &: (\texttt{user\_id}, \texttt{channel\_id}) \to (\texttt{vote}, \texttt{timestamp}) \\ \texttt{User} &: \texttt{user\_id} \to (\texttt{agreement\_count}, \texttt{total\_votes}, \texttt{created\_at})\end{aligned}$$

Additionally, the ML inference system maintains:

$$\begin{aligned}\texttt{ChannelFeatures} &: \texttt{channel\_id} \to (\texttt{feature\_vector}, \texttt{channel\_metadata}, \texttt{updated\_at}) \\ \texttt{Prediction} &: \texttt{channel\_id} \to (\texttt{is\_ai}, \texttt{confidence}, \texttt{model\_version})\end{aligned}$$

The blacklist is derived as $\{c : \texttt{status}(c) = \texttt{BLACKLISTED}\}$, where status transitions to `BLACKLISTED` when $\texttt{vote\_count\_ai} > \texttt{vote\_count\_human}$ and $\texttt{vote\_count\_ai} + \texttt{vote\_count\_human} \geq n_{\min}$.

5.3 Client Architecture

The Chrome extension operates in three modes:

(1) **Passive filtering:** Hides videos from blacklisted channels on YouTube home, search, and recommendation pages.

(2) **Active labeling:** Allows users to submit votes on channels they encounter.

(3) **Sync:** Periodically fetches updated blacklist from central server.

For unreviewed channels, the extension requests ML classification from the backend API and displays the prediction with confidence score, allowing users to make informed decisions before sufficient crowdsourced votes accumulate.

5.4 Backend Services

The backend is implemented as a FastAPI service with the following endpoints:

- `POST /api/v1/channels/analyze` — Full channel analysis with feature extraction and ML prediction
- `POST /api/v1/predict` — Direct prediction from pre-computed features
- `GET /api/v1/channels/blocked` — List channels flagged as AI-generated
- `GET /health` — Service health check with model status

The service uses Supabase as the persistence layer, with Row Level Security (RLS) policies enforcing data access controls.

6 Recommended Parameters

Based on theoretical analysis and practical considerations, we recommend the following MVP parameters:

Parameter	Value	Rationale
$n_{\min}$ (vote threshold)	5	Balances coverage with confidence; binomial conf. $> 95\%$ at $\alpha = 0.7$
r (rate limit)	50/day	Exceeds typical engagement; limits automated attacks
$\alpha_{\min}$ (accuracy floor)	0.3	Identifies adversarial behavior; allows honest disagreement
$n_{\text{calibration}}$	20	Sufficient samples to estimate α with reasonable precision
ML Classification		
Confidence threshold (block)	0.7	High-confidence predictions for automatic blocking
Confidence threshold (warn)	0.5	Medium-confidence predictions for user warning
Cache TTL	7 days	Balance freshness vs. API cost

7 Future Extensions

The MVP architecture admits several natural extensions as the user base grows:

Verified identity tier. Introduce optional OAuth authentication (Google, GitHub, etc.) for users who wish to have higher vote weights. This creates a two-tier system: frictionless participation for casual users, with stronger Sybil resistance for committed contributors.

Trust-weighted voting. Replace majority voting with weighted aggregation:

$$\hat{y}(c) = \mathbf{1}\left[\frac{\sum_{u \in V(c)} w(u) \cdot v(u,c)}{\sum_{u \in V(c)} w(u)} > 0.5\right] \tag{5}$$

where weights $w(u)$ derive from observed accuracy and verification status.

Bayesian aggregation. Jointly infer user accuracies and channel labels using expectation-maximization (Dawid-Skene model) or variational inference.

Active learning. Use ML model uncertainty to prioritize which channels to request human labels for, maximizing the information gain per vote.

Federated model submission. Allow users to submit classification models that compete based on held-out accuracy, with the best-performing model becoming the community default.

Additional feature sources. Incorporate video-level features (audio analysis, visual similarity detection, metadata patterns) to improve classification accuracy beyond channel-level statistics.

8 Conclusion

Clarity provides a minimal viable framework for crowdsourced AI content detection that combines machine learning classification with channel-level consensus voting. By using lightweight pseudonymous identity and simple majority aggregation, the system addresses the cold-start problem through ML predictions while building toward community-validated ground truth.

The current XGBoost classifier achieves 72% accuracy with 0.767 ROC-AUC on 3,217 labeled samples—sufficient for useful predictions, though far from definitive. The 21-feature model captures volume metrics, engagement ratios, and content metadata that show signal for distinguishing AI-generated channels from organic creators.

The framework's theoretical guarantees are modest: with honest majority participation, correct classifications emerge with high probability as vote counts grow, and attack costs scale linearly with community engagement. We have been explicit about the limitations—accuracy tracking circularity, vulnerability to strategic adversaries, and the inherent subjectivity of the "AI-generated" label. These are real constraints, not obstacles to be hand-waved away.

We offer this paper as a starting point, not a solution. The extensions in Section 7 sketch a path toward more robust mechanisms, but significant work remains. If you're interested in contributing—whether through code, theoretical improvements, or alternative approaches—we welcome collaboration.

References

[1] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794).

[2] Dawid, A. P., & Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1), 20–28.

[3] Douceur, J. R. (2002). The Sybil Attack. In *International Workshop on Peer-to-Peer Systems* (pp. 251–260). Springer.

[4] SponsorBlock. (2024). Crowdsourced YouTube sponsor segment database. `https://sponsor.ajay.app/`

[5] Raykar, V. C., et al. (2010). Learning from crowds. *Journal of Machine Learning Research*, 11, 1297–1322.