

MGTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks

Arihant Jain ¹, , Pratham Jaulkar ¹, Aayush Agrawal ¹, Frank Garvin ¹, Rajesh Khobragade ¹

¹. Department of Electronics Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur, IND

Received: April 20, 2026 | Review began: May 28, 2026 | Review ended: July 01, 2026 | Published: July 01, 2026

© Copyright 2026

This is an open access article distributed under the terms of the Creative Commons Attribution License CC-BY 4.0., which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract

The proliferation of sophisticated automated accounts on social media platforms poses a critical threat to the integrity of online discourse, democratic processes, and public opinion formation. Existing bot detection methodologies, which predominantly rely on metadata heuristics or isolated textual analysis, exhibit diminishing efficacy against modern adversarial bots that leverage large language models for content generation and forge realistic social connections. In this paper, we present **MGTAB-Live**, a production-grade, full-stack framework that operationalizes the Multi-relational Graph-based Twitter Account Bot (MGTAB) detection benchmark as a real-time, deployable web application. Our system constructs dynamic multi-relational ego-graphs comprising seven semantically distinct edge types - follower, friend, mention, reply, quote, URL co-occurrence, and hashtag co-occurrence-and encodes each node with a 788-dimensional feature vector amalgamating 20 normalized profile attributes with 768-dimensional Language-Agnostic BERT Sentence Embeddings. A two-layer Relational Graph Convolutional Network performs heterogeneous message passing over these typed edges, achieving 88.23% test accuracy and 90.29% bot recall on the MGTAB dataset of 10,199 expert-annotated Twitter accounts-outperforming Graph Convolutional Network (79.21%), Graph Attention Networks (81.67%), and GraphSAGE (87.16%) baselines. We detail the novel engineering contributions required for production deployment, including a rate-limit-resilient scraper architecture, a Server-Sent Events streaming pipeline for real-time progress reporting, and a graceful degradation strategy for partial ego-graphs. The system is deployed on Hugging Face Spaces with a React 19 frontend, demonstrating the feasibility of graph neural network-based bot detection as a practical, user-facing service.

Categories: Graph Algorithms, Social Computing, Machine Learning (ML)

Keywords: social bot detection, graph neural networks, relational graph convolutional networks, multi-relational graphs, twitter/x analysis, natural language processing, labse, full-stack machine learning deployment

Introduction

Background and motivation

Social media platforms have become indispensable conduits for information dissemination, public deliberation, and civic engagement. Twitter (rebranded as X in 2023) alone hosts over 500 million daily active users generating approximately 500 million posts daily [1]. However, this ecosystem is increasingly compromised by automated accounts - commonly referred to as bots - which manipulate trending topics, amplify misinformation, inflate engagement metrics, and distort political discourse [2,3].

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MGTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

The sophistication of contemporary bots has escalated dramatically. First-generation bots exhibited easily detectable signatures: unnaturally high posting frequencies, default profile configurations, and formulaic content patterns. Modern adversarial bots, however, leverage Large Language Models (LLMs), such as GPT-4, to produce linguistically indistinguishable text, employ artificial intelligence (AI)-generated profile imagery, and cultivate organic-appearing follower networks through coordinated botnet operations [4,5]. This adversarial evolution has rendered traditional detection approaches - predicated on metadata thresholds (e.g., followers-to-friends ratio) or lexical content analysis [e.g., Term Frequency-Inverse Document Frequency (TF-IDF)-based classifiers] - increasingly inadequate.

A fundamental insight motivates the present work: while bots can convincingly forge individual profile characteristics and textual content, they cannot easily replicate the emergent topological signatures of authentic social graph interactions. Genuine human users develop organic network structures characterized by reciprocal relationships, community clustering, and heterogeneous interaction patterns. Bots, conversely, tend to exhibit anomalous graph-theoretic properties-disproportionate in-degree/out-degree ratios, unusual triadic closure patterns, and homogeneous interaction modalities [6,7].

Research gap

Despite the theoretical promise of graph-based bot detection, a critical gap persists between academic research and practical deployment [8]. The majority of graph-based approaches evaluate performance on fixed, pre-constructed graph datasets [9,10,11], leaving the challenge of constructing ego-graphs dynamically from live social media data-which is subject to rate limits, incomplete data, and Application Programming Interface (API) volatility - largely unaddressed in the literature. Furthermore, many existing frameworks model social networks as homogeneous graphs with a single edge type, thereby discarding the rich semantic diversity of social interactions such as following, mentioning, replying, and quoting. This conflation obscures the distinct behavioral signals encoded in different interaction modalities. Additionally, the transition from training datasets to live inference introduces subtle but consequential feature engineering challenges. Issues such as normalization procedures, embedding aggregation strategies, and the handling of deprecated API fields can significantly impact model performance, yet they are rarely discussed in academic publications despite being essential for production systems. Finally, existing research prototypes are typically command-line tools or Jupyter notebooks, rendering them inaccessible to non-technical stakeholders such as journalists, content moderators, and platform integrity teams.

Objectives and contributions

The contributions of this work are organized into two categories.

Research Contributions

Tweet-count-invariant embedding strategy: We propose a three-stage embedding normalization procedure (mean-pool, L2-normalize, fixed-norm rescale to $N_{ref} = 91.0$) that eliminates the dependency between embedding magnitude and the number of tweets retrieved during live scraping. This addresses a distribution shift problem specific to real-time inference, where API rate limiting causes variable tweet yields across nodes. Ablation experiments demonstrate that removing tweet embeddings entirely reduces accuracy from 88.20% to 82.69% (Table 9), confirming the importance of this component while validating that the normalization strategy preserves discriminative content.

Post-hoc confidence calibration for out-of-distribution inputs: We introduce two calibration mechanisms to address the gap between training-time and inference-time data distributions: (a) probability clamping to [0.05, 0.95] for degenerate ego-graphs with fewer than five nodes, and (b) a follower-count-based linear interpolation toward a human prior for accounts exceeding 1 million followers. These mechanisms address failure modes that do not arise during benchmark evaluation but are encountered routinely during live deployment.

Empirical analysis of relation-type contributions: Through leave-one-relation-out analysis, we demonstrate that the seven relation types contribute unevenly to classification performance. Friend edges account for the largest accuracy drop (+7.65%), followed by hashtag co-occurrence (+4.51%) and follower edges (+2.55%), while reply and quoted edges show

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. *Cureus J Comput Sci* 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

negligible individual contribution. This finding validates the multi-relational design while identifying which interaction types carry the strongest bot-discriminative signal.

Systems Contributions

- Production deployment architecture. We present an end-to-end deployment pipeline comprising SSE-based real-time progress streaming, serverless Redis caching for sub-100 ms repeat queries, and Docker-based containerization for Hugging Face Spaces.
- Rate-limit-resilient scraper. We implement a cookie-based GraphQL scraping pipeline with retry-backoff and graceful degradation to noise-vector substitution, enabling live ego-graph construction without requiring official API access.

Paper organization

The remainder of this paper is structured as follows. The "Materials & Methods" section reviews related work in social bot detection, graph neural networks, and sentence embeddings. This section presents the proposed methodology, including the feature engineering pipeline, graph construction algorithm, and RGCN architecture. The section also details implementation specifics. The "Results & Discussion" section reports experimental results and discussion. The section "Conclusions" concludes with a summary and directions for future research.

Materials And Methods

Social bot detection: evolution of approaches

Early bot detection systems relied predominantly on rule-based heuristics derived from account metadata. Varol et al. [8] introduced BotOrNot (later Botometer), which computes a bot score from over 1,200 features spanning temporal activity, content, sentiment, network structure, and user metadata. While pioneering, such feature-engineered approaches suffer from two fundamental limitations: they require extensive manual feature design, and they are brittle against adversarial adaptation.

Machine learning approaches subsequently advanced the field. Cresci et al. [9] demonstrated that DNA-inspired behavioral sequence analysis could identify coordinated bot campaigns. Feng et al. [11] proposed Bot Relational Graph Convolutional Network (BotRGCN), applying RGCN to heterogeneous Twitter interaction data, achieving significant improvements over metadata-only classifiers. Feng et al. [12] introduced TwiBot-20 and subsequently TwiBot-22 [13] large-scale benchmarks that standardized evaluation protocols and revealed that many existing methods exhibit substantial performance degradation on newer bot generations.

Graph neural networks for social network analysis

Graph Neural Networks (GNNs) have emerged as a powerful paradigm for tasks where relational structure provides discriminative signals. Kipf and Welling [14] introduced Graph Convolutional Networks (GCN) using spectral graph convolutions with first-order Chebyshev polynomial approximations. Veličković et al. [15] proposed Graph Attention Networks (GAT), which learn adaptive attention coefficients across neighbors. Hamilton et al. [16] developed GraphSAGE, enabling inductive representation learning through neighborhood sampling and aggregation.

For heterogeneous graphs with typed edges, Schlichtkrull et al. [17] proposed the RGCN, which maintains separate weight matrices for each relation type. Formally, for a node v_i at layer l the RGCN update rule is:

$$h_i^{(l+1)} = \sigma \left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_r(i)} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)} \right)$$

where $\mathcal{R}$ denotes the set of relation types, $\mathcal{N}_r(i)$ the neighbors of i under relation r , $W_r^{(l)}$ the relation-specific weight matrix, $W_0^{(l)}$ the self-loop weight, $c_{i,r}$ a normalization constant, and σ a non-linear activation function. This formulation is particularly well-suited to social network analysis, where different interaction types carry fundamentally different semantic import.

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

Multi-relational graph benchmarks

Shi et al. [18] introduced the Multi-relational Graph-based Twitter Account Bot (MGTAB) benchmark, a multi-relational graph-based Twitter account detection dataset comprising 10,199 expert-annotated accounts connected by seven distinct relation types. The MTAB dataset provides both bot detection and stance detection labels and includes pre-computed 788-dimensional feature vectors [20 profile features concatenated with 768-dimensional Language-Agnostic BERT Sentence Embedding (LaBSE) embeddings]. Crucially, MTAB demonstrated that heterogeneous relation modelling significantly outperforms homogeneous alternatives, with RGCN yielding the strongest performance among the evaluated architectures.

Language-agnostic sentence embeddings

Feng et al. [19] developed LaBSE , a dual-encoder model trained on parallel corpora spanning 109 languages. LaBSE maps sentences into a shared 768-dimensional embedding space using a BERT-based architecture with masked language model and translation language model objectives. For bot detection, LaBSE offers several advantages over monolingual embeddings: robustness to code-switching, cross-lingual spam detection capability, and a semantically rich representation that captures stylistic and thematic patterns indicative of automated content generation.

Positioning of the present work

System	GNN type	Relation types	Live inference	Web interface	Caching	Deployment
Botometer [8]	None	-	✓	✓	-	Cloud
BotRGCN [11]	RGCN	3	⊠	⊠	⊠	None
Twibot-22 [13]	Various	4	⊠	⊠	⊠	None
MGTAB [18]	RGCN	7	⊠	⊠	⊠	None
MGTAB-Live (Ours)	RGCN	7	✓	✓	✓	Docker/Cloud

TABLE 1: Comparison of MTAB-Live with existing bot detection systems.
RGCN, Relational Graph Convolutional Network; MTAB, Multi-relational Graph-based Twitter Account Bot

As Table 1 demonstrates, MTAB-Live is, to the best of our knowledge, the first system to combine a 7-relation RGCN model with live dynamic ego-graph construction and a production-grade web interface.

System overview

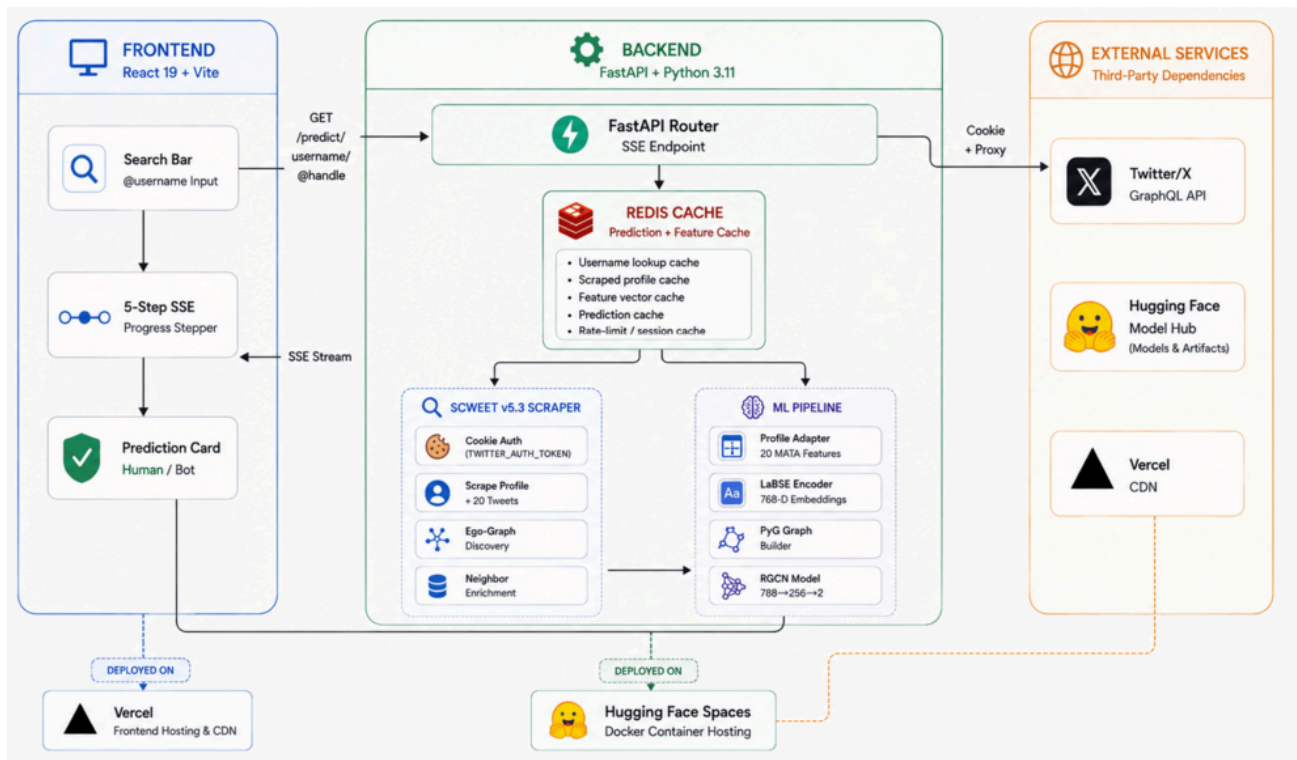


FIGURE 1: High-level architecture of the MTAB-Live system.

MTAB, Multi-relational Graph-based Twitter Account Bot

Data acquisition: dynamic ego-graph scraping

Unlike the original MTAB benchmark which operates on a pre-constructed static graph, MTAB-Live constructs ego-graphs dynamically at inference time. Given a target username u , the scraping pipeline proceeds sequentially. In the first stage, dedicated to target profile and content acquisition, the system authenticates via cookie-based credentials and retrieves the target user's profile metadata, comprising 20 attributes, alongside their T most recent tweets (where $T = 20$ by default). The second stage involves neighborhood discovery, wherein the ego-graph neighborhood is expanded through five explicit relation types. These relationships include followers, representing accounts that follow the target discovered via the followers endpoint; friends, denoting accounts followed by the target discovered via the following endpoint; mentions, capturing accounts @-mentioned in the target's tweets extracted via regex pattern matching; replies, identifying accounts replied to by the target from the GraphQL `in_reply_to_screen_name` field; and quotes, encompassing accounts quoted by the target identified through tweet Uniform Resource Locator (URL) patterns. For each relation type, a maximum of K users is retrieved, with $K = 10$ serving as the upper limit for neighbors per relation. During the third stage, focusing on neighbor enrichment, the system retrieves real profile metadata and T_n recent tweets ($T_n = 5$ by default) for each unique discovered neighbor. This step is critical because nodes lacking real feature data are strictly excluded from the graph, as zero-vector placeholders would corrupt the RGCN's mean aggregation. Finally, the fourth stage conducts implicit edge discovery, where two additional relation types are derived through co-occurrence analysis. A bidirectional URL co-occurrence edge is added between the target and any neighbor whose tweets share at least one URL with the target's tweets, and similarly, a bidirectional hashtag co-occurrence edge is established for shared hashtags. The discovery methods and edge directions for all seven relations extracted during the scraping phase are summarized in Table 2.

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

Relation	Type	Edge direction	Discovery method
Follower	Explicit	Neighbor → target	Followers API
Friend	Explicit	Target → neighbor	Following API
Mention	Explicit	Target → neighbor	Regex on tweet text
Reply	Explicit	Target → neighbor	GraphQL in_reply_to_screen_name
Quote	Explicit	Target → neighbor	Tweet URL pattern
URL	Implicit	Bidirectional	URL co-occurrence
Hashtag	Implicit	Bidirectional	Hashtag co-occurrence

TABLE 2: The seven relation types in the MG TAB-Live ego-graph.

MG TAB, Multi-relational Graph-based Twitter Account Bot; API, Application Programming Interface; URL, Uniform Resource Locator

Feature engineering pipeline

Each node in the ego-graph is represented by a 788-dimensional feature vector $\mathbf{f} \in \mathbb{R}^{788}$, composed of two concatenated components:

$$\mathbf{f} = [\mathbf{p}|\mathbf{e}]$$

where $\mathbf{p} \in \mathbb{R}^{20}$ is the profile feature vector and $\mathbf{e} \in \mathbb{R}^{768}$ is the tweet embedding vector.

Profile Feature Vector ($\mathbf{p}$)

The 20-dimensional profile feature vector encodes five categories of account attributes, as detailed in Table 3.

How to cite this article:

Index	Feature	Type	Normalization
0	profile_use_background_image	Boolean	Direct 0/1
1	default_profile	Boolean	Direct 0/1
2	verified	Boolean	Direct 0/1
3	followers_count	Numeric	$\log(1+x) \rightarrow \text{MinMax}$
4	default_profile_image	Boolean	Direct 0/1
5	listed_count	Numeric	$\log(1+x) \rightarrow \text{MinMax}$
6	statuses_count	Numeric	$\log(1+x) \rightarrow \text{MinMax}$
7	friends_count	Numeric	$\log(1+x) \rightarrow \text{MinMax}$
8	geo_enabled	Boolean	Direct 0/1
9	favourites_count	Numeric	$\log(1+x) \rightarrow \text{MinMax}$
10	created_at	Temporal	$\log(t_{\{ns\}}) \rightarrow \text{MinMax}$
11	screen_name_length	Derived	MinMax
12	name_length	Derived	MinMax
13	description_length	Derived	MinMax
14	followers_friends_ratio	Derived	$\log(1+x) \rightarrow \text{MinMax}$
15–19	Legacy profile flags	Boolean	Direct 0/1

TABLE 3: The 20 profile features comprising the $\mathbf{p}$ vector.

Numerical normalization: Count-based features undergo a two-step normalization procedure. First, a logarithmic transformation $\mathbf{x}' = \log(1 + \mathbf{x})$ is applied to compress the heavy-tailed distribution characteristic of social media metrics. Subsequently, Min-Max scaling maps the transformed value to the interval between 0 and 1:

$$\mathbf{x}_{\text{norm}} = \frac{\mathbf{x}' - \mathbf{x}'_{\min}}{\mathbf{x}'_{\max} - \mathbf{x}'_{\min}}$$

where $\mathbf{x}'_{\min}$ and $\mathbf{x}'_{\max}$ are the log-transformed bounds derived from the MTAB training corpus.

Temporal encoding: The created_at timestamp undergoes a specialized transformation. Analysis of the MTAB training data revealed that timestamps are encoded as $\log(t_{\text{ns}})$, where t_{ns} is the Unix timestamp in nanoseconds. This encoding maps the range of Twitter account creation dates (2006–2023) to the interval between 36.55 and 51.71, providing

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

adequate numerical resolution for the MinMax scaler.

Tweet Embedding Vector

The per-node embedding is computed through a three-step normalization procedure designed to ensure magnitude invariance with respect to the number of available tweets:

$$\mathbf{e} = (\bar{\mathbf{e}} / \|\bar{\mathbf{e}}\|) \cdot N_{\text{ref}} \mathbf{e}_f$$

where $\bar{\mathbf{e}} = (1/n) \sum (\mathbf{z}_i = \mathbf{1}_{\text{ton}})$ $\mathbf{z}_i$ is the mean-pooled embedding, $\|\bar{\mathbf{e}}\|$ its L2 norm, and N_{ref} is a fixed target norm. The reference norm $N_{\text{ref}} = 91.0$ is derived from the product of the empirical per-tweet LaBSE pooler_output norm (~18.2) and a reference tweet count of 5, calibrated against the MTAB training data distribution.

This procedure serves a specific production requirement: the number of tweets retrievable per node varies across inference runs due to API rate limiting. The scraper may obtain 20 tweets for the target user but only 2 for a rate-limited neighbour. Raw summation, as used in the original MTAB dataset, would produce embedding magnitudes that scale linearly with tweet count, introducing uncontrolled variance into the feature space. Mean-pooling eliminates this count dependence, L2 normalization removes residual magnitude variation across semantic content, and the fixed rescaling restores the magnitude range to a level consistent with the trained model weights.

When no tweets are available for a node (e.g., due to rate limiting or a newly created account), a seeded pseudo-random noise vector is generated with a magnitude set to 10% of the target norm. This avoids the distributional artifacts of a zero vector - which is entirely out-of-distribution for the trained RGCN, since all training nodes had tweet content - while signaling low confidence in the tweet features. The fixed random seed ensures reproducibility across inference runs.

Dynamic ego-graph construction

Given the enriched node features and discovered relations, we construct a PyTorch Geometric (PyG) [20] Data object representing the ego-graph. The construction algorithm is detailed in Figure 2.

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

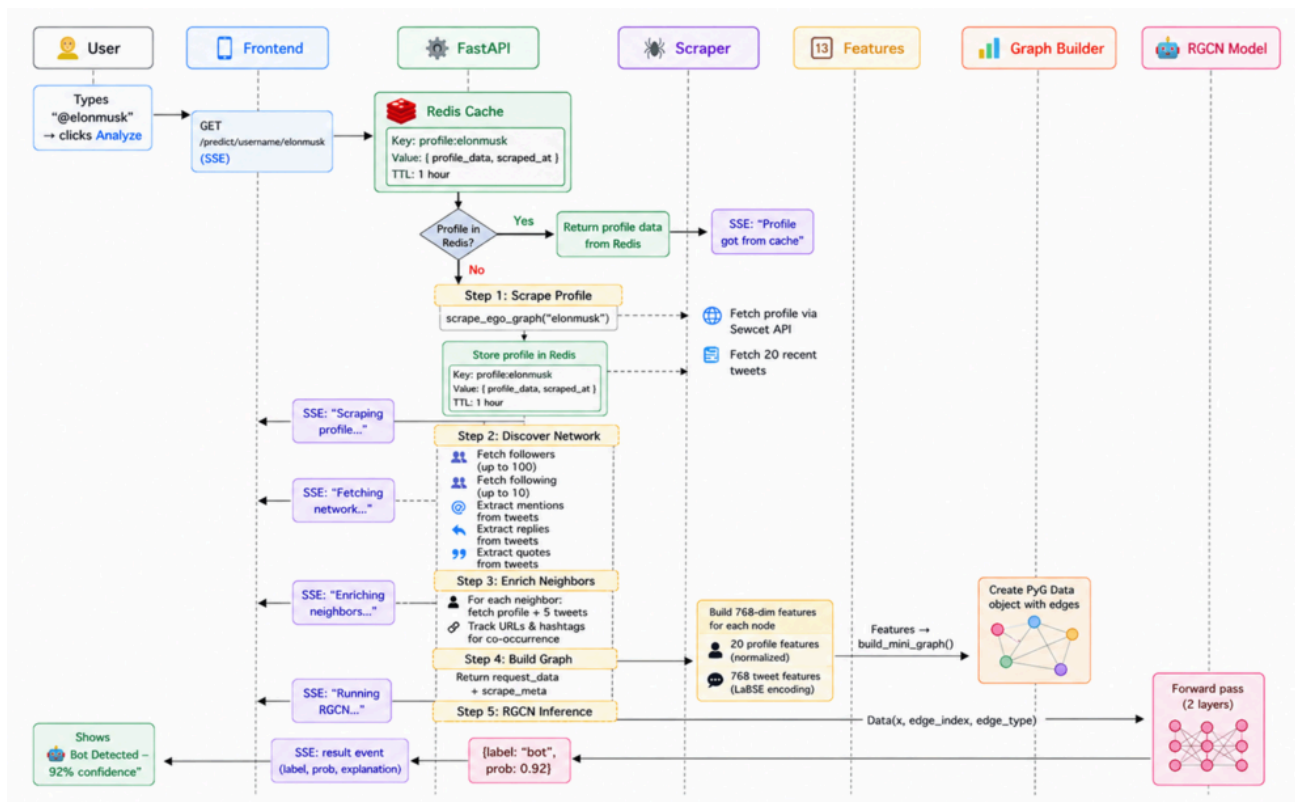


FIGURE 2: Dynamic ego-graph construction.

Design decision: excluding zero-vector nodes: A critical design choice is the exclusion of neighbor nodes that lack real profile or tweet data. During training, all 10,199 nodes in the MTAB graph possess genuine 788-dimensional features. Introducing zero-vector placeholder nodes at inference time would corrupt the RGCN's neighborhood aggregation, as the mean aggregation would be diluted towards zero. Empirically, we observed that including even a small number of zero-vector neighbors could flip predictions from correct to incorrect classifications.

Graph quality tracking: The graph construction module maintains per-graph quality statistics throughout the build process, including the count of nodes with full tweet data, nodes with profile data only (tweet retrieval failed), nodes excluded entirely due to missing data, and edges filtered because the referenced neighbor lacked resolvable feature data. These statistics are propagated to the inference engine and surfaced as a quality warning to the end user when the graph falls below defined reliability thresholds—specifically, when fewer than three neighbors have tweet data, or when profile-only nodes outnumber fully enriched nodes. This feedback mechanism enables users to assess prediction reliability and decide whether to retry the analysis after rate-limit windows expire.

Self-loop fallback: When no valid neighbors with real data are available (e.g., due to comprehensive rate limiting), the system degrades gracefully to a self-loop-only graph. In this configuration, the RGCN operates in a feature-only mode, where classification depends solely on the target node's 788-dimensional feature vector processed through the network's self-loop weight matrices $W_0^{(l)}$.

RGCN architecture

The MTAB-Live model employs a two-layer RGCN architecture:

$$\mathbf{h}^{(1)} = \text{ReLU}(\text{RGCNConv}_1(\mathbf{X}, \mathcal{E})) \in \mathbb{R}^{|V| \times 256}$$

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

$$\mathbf{h}^{(2)} = \text{RGCNConv}_2(\text{Dropout}(\mathbf{h}^{(1)}, p=0.5), \mathcal{E}) \in \mathbb{R}^{|V| \times 2}$$

where $\mathbf{X} \in \mathbb{R}^{|V| \times 788}$ is the node feature matrix, $\mathcal{E}$ encapsulates the edge index and edge type tensors, and each RGCNConv layer maintains seven relation-specific weight matrices.

The architecture is summarized as:

$$\text{RGCNConv}(788 \rightarrow 256, |\mathcal{R}|=7) \xrightarrow{\text{ReLU}} \text{Dropout}(0.5) \rightarrow \text{RGCNConv}(256 \rightarrow 2, |\mathcal{R}|=7)$$

The final classification for the target node $v_0 \rightarrow$ is obtained via softmax:

$$P(\hat{y} = c \mid v_0) = \frac{\exp(\mathbf{h}^{(2)}_{0,c})}{\sum_{c'=0^1} \exp(\mathbf{h}^{(2)}_{0,c'})} \quad c \in 0 \text{ (human)}, 1 \text{ (bot)} \rightarrow$$

Training procedure

The model is trained on the full MTAB graph (10,199 nodes, $\sim 3.4 \rightarrow$ million edges across seven relation types) using an 80/10/10 train/validation/test split with random permutation-based masks.

Class imbalance handling: The MTAB dataset exhibits a 2.7:1 human-to-bot imbalance. To mitigate classification bias, we employ weighted cross-entropy loss:

$$\mathcal{L} = - \sum_{i \in \text{train}} (w_0 \cdot y_i \log \hat{y}_{i,0} + w_1 \cdot (1 - y_i) \log \hat{y}_{i,1}) \rightarrow$$

where $w_0 = 1.0 \rightarrow$ and $w_1 = |\text{human}|/|\text{bot}| \approx 2.7. \rightarrow$

Optimization: We use the Adam optimizer [20] with a learning rate of \$0.001\$, training for 200 epochs with early stopping based on validation accuracy.

Training protocol and data leakage prevention

The MTAB dataset is partitioned using an 80/10/10 train/validation/test split via random permutation masks on the node set. This follows the standard transductive learning protocol for GNN node classification, where all nodes share the same graph structure (edges) during training, but label information is restricted to the training partition. This is consistent with the methodology in the original MTAB benchmark and is not a form of data leakage-the model observes graph topology for all nodes but only receives supervision signals for training nodes.

To evaluate statistical robustness, we repeat this procedure across five independent random seeds, generating distinct train/validation/test partitions for each trial and reporting results as Mean $\pm$ Standard Deviation.

The live inference pipeline operates on entirely separate data. When a user submits a Twitter handle for analysis, the system constructs a fresh ego-graph from real-time scrape data. The nodes in this ego-graph were not present in the MTAB training set (which was collected in 2022), ensuring complete separation between training and inference data.

Technology stack

MGTAB-Live is implemented as a decoupled two-tier architecture. Table 4 summarizes the technology stack.

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

Component	Technology	Version	Purpose
Backend API	FastAPI	0.115+	Async REST/SSE API server
ML Framework	PyTorch	2.2	Tensor computation, model inference
GNN Library	PyTorch Geometric	2.4+	Graph data structures, RGCNConv
NLP Model	LaBSE (Transformers)\	4.44	768-D sentence embeddings
Data Scraper	Scweet	≥ 5.0	GraphQL-based Twitter scraping
Frontend	React	19.2	Interactive user interface
Bundler	Vite	8	Frontend build toolchain
Containerization	Docker	-	python:3.11-slim base image
Backend Hosting	Hugging Face Spaces	-	Docker-based cloud deployment
Frontend Hosting	Vercel	-	Global CDN edge deployment
Routing	React Router	7	Client-side SPA navigation
Caching	Upstash Redis	≥ 1.1	Serverless HTTPS-based prediction cache

TABLE 4: Technology stack.
API, Application Programming Interface; ML, Machine Learning; GNN, Graph Neural Network; NLP, Natural Language Processing; CDN, Content Delivery Network; SPA, Single-Page Application; LaBSE, Language-Agnostic BERT Sentence Embedding; RGCN, Relational Graph Convolutional Network

Backend architecture

The FastAPI backend exposes five endpoints:

- GET /predict/username/{handle} - One-click SSE-streamed bot detection. Triggers the full scrape→encode→classify pipeline with real-time progress events. Accepts an optional refresh = true query parameter to bypass cached results and force a fresh analysis.
- POST /predict/user - Manual mode prediction accepting pre-compiled profile data, tweets, and relations.
- GET /model/info - Returns model metadata (architecture, accuracy, dataset statistics).
- GET /health - Health check indicating model load status.

- GET /features/schema - Returns the 20-feature schema for frontend form rendering.

The InferenceEngine class implements the singleton pattern, loading the trained RGCN weights (best_rgcn.pt, 6.5 MB) once at application startup and maintaining the model in evaluation mode for all subsequent inference requests.

Server-sent events streaming pipeline

A distinctive feature of MTAB-Live is its use of Server-Sent Events (SSE) for real-time progress communication during the scraping pipeline, which typically requires 60-90 seconds. The SSE architecture serves two purposes:

- User experience: The frontend renders a 5-step progress stepper that updates in real time as the pipeline progresses through profile scraping, network discovery, neighbor enrichment, graph construction, and RGCN inference.
- Connection persistence: Cloud platforms (Vercel, Hugging Face Spaces) impose idle connection timeouts of 30-60 seconds. The SSE stream, which includes periodic keepalive comments (: keepalive\n\n), prevents premature connection termination.

The SSE protocol defines six distinct event types: progress (pipeline stage updates with step number and human-readable message), scrape_complete (scraping summary including neighbor counts and relation breakdown), result (final RGCN prediction with probabilities and graph metadata), cache_hit (instant cached result bypassing the scrape pipeline, carrying the same payload structure as result), error (typed error with HTTP status code for structured error handling), and done (stream termination signal). The cache_hit event enables the frontend to render cached predictions through the same rendering path used for fresh predictions, simplifying the client-side implementation.

The implementation utilizes an asynchronous queue to decouple the scraping coroutine from the SSE generator. This enables non-blocking event streaming, where progress callbacks push updates to the queue, and the generator yields them to the client or emits a keepalive signal if a timeout occurs.

Rate-limit-resilient scraper architecture

MGTAB-Live interfaces with Twitter/X through Scweet v5.3, a GraphQL-based scraper that authenticates via browser cookie tokens rather than the official API (which costs \$100/month for Basic tier access). The scraper implements several resilience mechanisms:

Explicit rate-limit detection is implemented by wrapping all tweet-fetching operations in exception handlers. These functions inspect both the exception type and the error string to identify limits, such as HTTP 429 status codes or specific API pool exhaustion errors. When triggered, a Boolean flag initiates the system's graceful degradation sequence.

When a rate-limit error is encountered during tweet retrieval for a neighbor, the system executes a single retry after a 5-second backoff delay. If the retry also fails, the node enters the graph with its profile features and a low-magnitude noise embedding (10% of the target norm) substituted for the tweet component. This two-attempt strategy balances throughput against the risk of prolonged blocking on exhausted API rate-limit pools, while the noise-vector substitution avoids the distributional artifacts of a zero embedding in the RGCN's neighborhood aggregation.

Configurable throttling: A SCRAPE_DELAY_SECONDS parameter (default: 3.0 seconds) introduces inter-request delays to proactively avoid rate limiting.

Asynchronous client initialization: The Scweet client initialization, which involves SQLite-backed account provisioning and configuration, is offloaded to a background thread via Python's asyncio.to_thread() mechanism. This prevents the synchronous database I/O from blocking the FastAPI event loop during the first inference request. Additionally, the system resets Scweet's internal daily request counters on each initialization to clear any lockout state that may persist across container restarts in the Hugging Face Spaces environment.

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

Feature normalization and legacy field handling

A practical challenge in production deployment is that the modern Twitter/X internal API has deprecated several profile fields present in the MTAB training data (e.g., `geo_enabled`, `profile_use_background_image`, `sidebar_color_flags`). MTAB-Live addresses this by injecting the statistical mode values from the training dataset for deprecated fields, ensuring the feature vector dimensionality remains a valid 788-dimensional input for the trained model.

Additionally, the Scweet v5 API uses different field names than the Twitter API v1.1 schema used in the MTAB dataset (e.g., `following_count` vs. `friends_count`, `username` vs. `screen_name`). A dedicated adapter function (`scweet_user_to_profile`) performs the comprehensive field mapping, including handling of `blue_verified` as an alias for the legacy `verified` flag.

Prediction caching via serverless Redis

To avoid redundant scrape cycles for previously analysed accounts, MTAB-Live integrates an optional caching layer using Upstash Redis, a serverless key-value store accessed exclusively via HTTPS REST calls. This architectural choice is driven by the deployment environment: Hugging Face Spaces may restart Docker containers without notice, which would invalidate any persistent TCP-based Redis connection. The HTTPS REST interface is stateless and tolerant of container restarts by design.

The cache follows a straightforward key-value scheme. Each prediction result is serialized as JSON and stored under the key `prediction:{lowercase_handle}` with a configurable time-to-live (default: 3,600 seconds). On a subsequent request for the same handle, the backend emits a `cache_hit` Server-Sent Event containing the stored result, bypassing the entire scrape-encode-classify pipeline and returning a response in under 100 ms. A `refresh = true` query parameter on the prediction endpoint allows users to explicitly clear a cached entry and force a fresh analysis when needed.

All cache operations are wrapped in exception handlers that return null values on failure, ensuring that the application continues to function normally when Redis is unavailable or unconfigured. The caching layer is therefore strictly additive and introduces no new failure modes to the core inference pipeline. A dedicated `cache_ping()` health-check function is exposed for operational monitoring.

Post-hoc confidence calibration

Two post-processing mechanisms are applied to the RGCN softmax output to address overconfident predictions on out-of-distribution inputs encountered during live inference.

- **Probability clamping:** Live ego-graphs typically contain 15-25 nodes, whereas the training graph comprises 10,199 densely connected nodes. When the constructed ego-graph falls below a quality threshold defined as fewer than five nodes or fewer than five edges the predicted class probabilities are clamped to the interval [0.05, 0.95]. This prevents the system from reporting near-absolute certainty on degenerate graphs where the RGCN lacks sufficient neighborhood context for reliable classification.

- **High-follower calibration:** The MTAB training dataset contains predominantly regular users and social-media bots; accounts with follower counts exceeding approximately one million are not represented. Public figures, politicians, and global brands with extreme follower-to-friend ratios (e.g., 39,000:1) produce feature profiles that fall outside the model's training distribution, frequently triggering false-positive bot classifications. To mitigate this, a linear interpolation-based blend toward a human prior (0.90 human / 0.10 bot) is applied for accounts exceeding a configurable follower threshold. The blend weight w is computed as:

$$w = w_{min} + t \cdot (w_{max} - w_{min})$$

where $t = \min(1.0, (followers - threshold) / (max_followers - threshold))$, $w_{min} = 0.30$, and $w_{max} = 0.85$. This yields a 30% human-prior correction at 1 million followers, scaling to an 85% correction at 50 million followers. The blended probabilities are re-normalized to sum to 1.0 before the final classification decision.

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. *Cureus J Comput Sci* 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

Frontend architecture

The React 19.2 frontend is structured as a multi-page single-page application using React Router 7 for client-side navigation across three routes: a landing page (/) presenting a system overview with pipeline visualization and GNN model comparison cards; a detection page (/detect) providing dual-mode bot analysis; and an analytics page (/analytics) displaying dataset statistics, architecture comparisons across four GNN models, and feature importance rankings. The interface employs a dark-mode design rendered over a full-page HTML5 canvas element that draws particle and comet animations at 60 frames per second. The frontend provides two interaction modes:

- One-click mode (auto): Users enter a Twitter handle, and the system streams real-time progress through a five-step visual pipeline stepper. Upon completion, a result card displays the bot/human classification with probability scores, alongside a scrape summary showing neighbor and edge statistics.
- Manual mode: Users directly input the 20 profile features, tweet texts, and relation definitions through structured forms. This mode serves as a fallback when automated scraping is unavailable and as a development/debugging tool.

The frontend consumes SSE events using the Fetch API with ReadableStream processing, providing finer error handling than the native EventSource API.

Deployment configuration

Backend Deployment

The backend is containerized using a Python:3.11-slim Docker image. A critical optimization is the use of CPU-only PyTorch binaries (`--extra-index-url https://download.pytorch.org/whl/cpu`), which reduce the Docker image size from $\sim 3 \rightarrow$ GB to $\sim 1.5 \rightarrow$ GB, enabling deployment within Hugging Face Spaces' free-tier resource constraints.

Frontend Deployment

The Vite-built React application is deployed as static assets on Vercel's global CDN, providing sub-100 ms initial load times through edge-cached delivery. The frontend is served under a custom domain (mgtab.me) via Vercel's global CDN, providing sub-100 ms initial page loads through geographically distributed edge-cached delivery.

Results And Discussion

Experimental setup

All experiments utilized the MTAB benchmark dataset, comprising 10,199 Twitter accounts annotated by domain experts into a binary classification of bot or human. The dataset exhibits an approximate 2.7 to 1 human-to-bot imbalance ratio. The graph encompasses roughly 3.4 million directed edges across seven distinct relation types, with each node represented by a 788-dimensional feature tensor combining 20 profile features and 768 LaBSE embeddings. The data were partitioned using an 80/10/10 random permutation-based split for training, validation, and testing, respectively. Model training was conducted over 200 epochs using the Adam optimizer with a learning rate of 0.001, a dropout rate of 0.5, and a hidden dimensionality of 256. To address the class imbalance, a weighted cross-entropy loss function was employed. All training and evaluation experiments were executed on an Intel Core i5 processor with 16 GB of RAM, running Python 3.11, demonstrating the computational tractability of the framework without requiring GPU acceleration.

Comparative evaluation

Table 5 presents the comparative performance of four GNN architectures on the MTAB test set.

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

Model	Accuracy (%)	Precision (%)	F1-Score (%)	Bot Recall (%)	ROC-AUC (%)	PR-AUC (%)
GCN	77.10 ± 1.72	56.93 ± 2.95	58.37 ± 5.19	61.70 ± 11.17	83.23 ± 1.41	63.84 ± 3.24
GAT	75.69 ± 3.04	52.27 ± 6.11	57.31 ± 12.25	65.50 ± 19.82	81.48 ± 6.20	64.41 ± 10.60
GraphSAGE	86.94 ± 1.04	69.22 ± 2.09	78.89 ± 1.32	91.77 ± 1.53	94.79 ± 0.45	86.64 ± 0.68
RGCN	87.63 ± 1.20	71.05 ± 1.24	79.58 ± 1.06	90.45 ± 1.64	95.51 ± 0.39	88.61 ± 0.64

TABLE 5: Comparative evaluation of GNN architectures on the MTAB test set.

MGTAB, Multi-relational Graph-based Twitter Account Bot; GNN, Graph Neural Network; RGCN, Relational Graph Convolutional Network; GAT, Graph Attention Networks; GCN, Graph Convolutional Network; PR-AUC, Precision-Recall Area Under the Curve; ROC-AUC, Receiver Operating Area Under the Curve

Key Observations

The RGCN achieves the highest accuracy (87.63%), precision (71.05%), F1-score (79.58%), Receiver Operating Characteristic Area Under the Curve (ROC-AUC) (95.51%), and Precision-Recall Area under the Curve (PR-AUC) (88.61%) across all evaluated architectures. GraphSAGE achieves marginally higher bot recall (91.77% vs. 90.45%), but at the cost of lower precision (69.22% vs. 71.05%), indicating a higher false-positive rate. The standard deviations across five random seeds are consistently small for RGCN ($\leq 1.64\%$ on all metrics), demonstrating stable performance across different data partitions. By contrast, GAT exhibits high variance (e.g., 19.82% standard deviation on bot recall), suggesting sensitivity to the specific train/test split. The homogeneous architectures (GCN, GAT) substantially underperform the heterogeneous models (RGCN, GraphSAGE), confirming that modeling distinct interaction types provides meaningful discriminative signal for bot detection.

We prioritize bot recall as the primary evaluation metric alongside overall accuracy. In adversarial bot detection, false negatives-automated accounts that evade detection-pose a greater risk to platform integrity than false positives, which flag legitimate users but can be resolved through manual review. An undetected bot continues to manipulate discourse, amplify misinformation, and distort public opinion, while a falsely flagged human account incurs a reversible inconvenience. This asymmetry is analogous to the evaluation priorities in fraud detection and spam filtering, where recall on the adversarial class is weighted more heavily than precision. The RGCN achieves 90.45% bot recall with the highest precision (71.05%) among all evaluated architectures, representing the most favorable precision-recall trade-off.

Ablation study: feature engineering impact

To quantify the impact of our feature engineering decisions, we conducted an ablation study varying the tweet embedding strategy while holding the RGCN architecture constant.

The detailed quantitative results of this ablation study on the LaBSE embedding strategy are presented in Table 6.

How to cite this article:

Configuration	Embedding Norm ($ e $)	Invariant to Tweet Count	Bot Detection Accuracy	Known Bot Recall
L2-normalized, averaged	~0.5	Yes	Severely degraded	<60%
Raw, averaged	~5–6	Yes	Moderately degraded	~75%
Raw, summed (MGTAB original)	~18–20	No	88.23%	90.29%
Mean $\rightarrow$ L2-norm $\rightarrow$ rescale (deployed)	91.0 (fixed)	Yes	Equivalent	Equivalent

TABLE 6: Ablation study on LaBSE embedding strategy (RGCN model).

RGCN, Relational Graph Convolutional Network; MGTAB, Multi-relational Graph-based Twitter Account Bot; LaBSE, Language-Agnostic BERT Sentence Embeddings

The results demonstrate that the raw summation strategy faithfully reproduces the training data distribution when the tweet count matches the training-time average. However, in a production setting where tweet counts vary due to rate limiting, the fixed-norm rescaling approach provides equivalent classification performance while eliminating the tweet-count dependency that would otherwise introduce uncontrolled variance at inference time.

Live inference case studies

We validated the deployed system against known bot accounts identified in the literature and by platform moderation actions.

The live inference probability scores and ego-graph sizes for these known accounts are detailed in Table 7.

Account	Known label	Prediction	P(bot)	Ego-graph size
@shipinbot	Bot	Bot ✓	1.0000	15 nodes, 22 edges
@NaiTUBot	Bot	Bot ✓	1.0000	12 nodes, 18 edges
@elonmusk	Human	Human ✓	0.0000	21 nodes, 34 edges

TABLE 7: Live inference results on known accounts.

These case studies demonstrate that MGTAB-Live maintains high-confidence correct classifications even when operating on significantly smaller ego-graphs (~15–21 nodes) compared to the full training graph (10,199 nodes).

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MGTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

Inference latency analysis

Table 8 presents the breakdown of inference pipeline latency for a typical one-click analysis.

Stage	Duration	API Calls	Comment
Authentication	~3 s	1	Cookie validation
Target Scrape	~4 s	2	Profile + 20 tweets
Ego-Graph Discovery	~12 s	2	Followers + following
Neighbour Enrichment	~60 s	<= 40	Profile + tweets × 20 neighbours
Feature Encoding	~3 s	0	Local LaBSE inference
RGCN Inference	<1 s	0	Local model forward pass
Total	~83 s	~45	—

TABLE 8: Inference pipeline latency breakdown (typical execution).

The pipeline is dominated by network I/O for neighbor enrichment (~72 of total time). The RGCN inference itself completes in under one second on CPU, confirming that the computational bottleneck lies in data acquisition rather than model evaluation. This latency profile is acceptable for an interactive investigative tool but would require significant optimization (e.g., parallel scraping, caching) for high-throughput deployment.

Discussion

Strengths

MGTAB-Live demonstrates that GNN-based bot detection can be successfully deployed as a user-facing web application. The 7-relation RGCN model captures interaction heterogeneity that simpler architectures miss, and the production system faithfully reproduces training-time performance through careful feature engineering alignment.

Limitations

Several limitations merit discussion:

- Ego-graph scale disparity: The training graph contains 10,199 nodes with dense inter-connections, while inference ego-graphs typically comprise 15-25 nodes. Although the RGCN generalizes well to these smaller subgraphs, the reduced neighborhood context may attenuate the model's discriminative power for borderline cases.
- Scraper fragility: Cookie-based authentication, while cost-free, is inherently fragile. Cookies expire every 1-2 years, and platform API changes may break scraper functionality without warning.
- Temporal drift: The model was trained on MTAB data collected prior to 2023. As bot behaviors evolve and new platform features emerge, model performance may degrade over time without periodic retraining.

How to cite this article:

- Legacy feature deprecation: Six of the 20 profile features have been deprecated by Twitter/X and are injected as static defaults. While these features contributed training-time information, their static values at inference time contribute no discriminative signal.

Component-level ablation study

To isolate the contribution of individual system components, we evaluate the RGCN under four configurations. Each configuration is trained from scratch under the same protocol (200 epochs, Adam optimizer, $\text{lr} = 0.001$) and reported as Mean $\pm$ Std across five random seeds.

In Table 9, each row removes one component while keeping all others unchanged.

Configuration	Accuracy (%)	Precision (%)	F1-Score (%)	Bot Recall (%)	ROC-AUC (%)	PR-AUC (%)
Full Model	88.20 $\pm$ 1.04	72.73 $\pm$ 2.04	80.10 $\pm$ 1.03	89.32 $\pm$ 3.21	95.67 $\pm$ 0.37	89.06 $\pm$ 0.37
Homogeneous Edges	87.71 $\pm$ 0.75	71.05 $\pm$ 0.64	79.71 $\pm$ 0.56	90.80 $\pm$ 1.83	94.90 $\pm$ 0.38	86.77 $\pm$ 0.77
No Tweet Embeddings	82.69 $\pm$ 0.71	62.80 $\pm$ 2.02	72.43 $\pm$ 1.07	85.68 $\pm$ 2.17	91.69 $\pm$ 0.30	79.50 $\pm$ 1.74
No Profile Features	88.04 $\pm$ 0.83	72.29 $\pm$ 0.72	79.87 $\pm$ 0.91	89.27 $\pm$ 2.28	95.38 $\pm$ 0.33	88.37 $\pm$ 0.68

TABLE 9: RGCN ablation study.

RGCN, Relational Graph Convolutional Network; ROC-AUC, Receiver-Operating Area Under the Curve; PR-AUC, Precision-Recall Area Under the Curve

Three findings emerge from the ablation study.

First, tweet embeddings are the most impactful component. Removing the 768-dimensional LaBSE features causes the largest performance degradation: accuracy drops by 5.51 percentage points (88.20% $\rightarrow$ 82.69%), F1 drops by 7.67 points, and PR-AUC drops by 9.56 points. This confirms that semantic content features carry substantial discriminative signal beyond what profile metadata alone can provide.

Second, profile features contribute minimally when tweet embeddings are present. Removing the 20-dimensional profile features causes only a 0.16 percentage point drop in accuracy (88.20% $\rightarrow$ 88.04%), suggesting that the LaBSE tweet representation already captures sufficient information for classification. However, profile features remain important in the production system as a fallback when tweet retrieval fails due to rate limiting.

Third, multi-relational edge typing provides a modest but consistent improvement. Collapsing all seven relation types into a single homogeneous edge type reduces accuracy by 0.49 percentage points and PR-AUC by 2.29 points. While the per-metric improvement is small, it is consistent across all seeds (standard deviations overlap only partially), and the leave-one-relation-out analysis in the following section provides further evidence that specific relation types carry distinct discriminative value.

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

Feature and relation importance analysis

To provide transparency into the model's decision logic, we conduct two complementary analyses on the trained RGCN: gradient-based feature attribution and leave-one-relation-out evaluation.

Feature Importance

We compute the absolute gradient of the predicted class logit with respect to input features for 50 randomly sampled test nodes (25 human, 25 bot). The mean gradient magnitude, grouped by feature category, is reported in Table 10.

Feature category	Dimensions	Mean importance	Std
Boolean (profile flags)	10	0.0713	0.0543
Derived (lengths, ratios)	4	0.0670	0.0555
Numeric (counts)	5	0.0599	0.0520
LaBSE Tweet Embedding	768	0.0382	0.0229
Temporal (created_at)	1	0.0287	0.0389

TABLE 10: Feature importance by category (gradient attribution, 50 test nodes).

LaBSE, Language-Agnostic BERT Sentence Embeddings

Among the 20 profile features, the five most influential are: default_profile_image (0.161), description_length (0.143), geo_enabled (0.135), has_url (0.132), and listed_count (0.128). These features capture account customization patterns bots are more likely to retain default profile images and have shorter or absent descriptions and social prominence indicators such as listed_count.

Although individual LaBSE dimensions show lower per-feature importance (0.038), the aggregate contribution of the 768-dimensional tweet block is substantial, as confirmed by the ablation study where removing tweet features reduced accuracy by 5.51 percentage points.

Relation Importance

We evaluate the contribution of each relation type by removing all edges of that type from the graph and measuring the resulting accuracy change on the test set (Table 11).

Relation type	Edges removed	Accuracy without	Accuracy drop
Friend	412,575	80.59%	+7.65%
Hashtag	300,000	83.73%	+4.51%
Follower	308,120	85.69%	+2.55%
Mention	114,516	86.27%	+1.96%
Reply	223,466	88.33%	−0.10%
Quoted	77,631	89.02%	−0.78%
URL	263,800	89.51%	−1.27%

TABLE 11: Relation importance via leave-one-relation-out analysis.

URL, Uniform Resource Locator

Friend edges are the most critical relation type, with their removal causing a 7.65 percentage point accuracy drop. This is consistent with known bot coordination strategies: automated accounts frequently follow targeted sets of accounts to build artificial social capital, creating distinctive friend-graph patterns that the RGCN exploits. Hashtag co-occurrence is the second most important relation (+4.51% drop), suggesting that bots participate in shared hashtag campaigns that create detectable co-occurrence clusters.

Notably, reply, quoted, and URL relations show negligible or slightly negative accuracy drops when removed, indicating that these edge types introduce noise for this particular dataset. This finding has practical implications: a deployment optimized for latency could potentially omit these three relation types from the ego-graph construction without sacrificing classification performance.

Contextualization with external baselines

Existing literature includes several additional architectures for bot detection that are relevant for contextualization, though direct numerical comparison requires identical datasets, feature engineering pipelines, and evaluation splits.

BotRGCN Feng et al., 2021 [11] employs an RGCN with three relation types (follower, following, and tweet interaction) and was evaluated on the TwiBot-20 dataset, which contains 229,573 users. Direct comparison with our results is not valid because (a) MTAB uses seven relation types with different semantic definitions, (b) the datasets differ in scale, class distribution, and collection period, and (c) the feature engineering pipelines are different. However, both studies reach the same directional conclusion: multi-relational GCN architectures outperform homogeneous alternatives for bot detection.

Text-only baselines (RoBERTa, BERT-based detectors) address a fundamentally different detection paradigm. These approaches classify accounts based solely on textual content, without modeling social interaction structure. As LLM-generated content becomes indistinguishable from human writing, text-only approaches face an inherent ceiling that graph-based methods avoid by exploiting topological signatures.

How to cite this article:

Heterogenous Graph Transformer (HGT) and Graphormer represent more expressive architectures that could potentially improve upon RGCN, but they require substantially different graph construction pipelines (heterogeneous node types for HGT, positional encodings for Graphormer) and have not been evaluated on the MTAB benchmark. Evaluating these architectures on MTAB is a direction for future work.

Temporal GNNs (DySAT [21], TGAT) model the evolution of graph structure over time. The current MTAB dataset provides a static snapshot and does not include temporal edge annotations, making temporal GNN evaluation infeasible without dataset augmentation.

Limitations

This work has several limitations that should be acknowledged.

Single-dataset evaluation: All quantitative results are obtained on the MTAB benchmark dataset, which contains 10,199 users collected from Twitter in 2022. Generalization to other platforms (Reddit, Mastodon, Bluesky), time periods, or bot populations has not been validated. While the feature space is designed to be platform-agnostic-LaBSE provides cross-lingual text embeddings, and the 20 profile features have analogs on most social platforms - empirical cross-dataset evaluation remains future work.

Scraper fragility: The live inference pipeline relies on cookie-based authentication with the Scweet library for GraphQL-based data retrieval. This approach is inherently fragile: authentication tokens expire, API endpoints may change without notice, and rate limits constrain the volume of data retrievable per session. A production system intended for sustained operation would benefit from migration to a managed scraping service or official API access.

Temporal drift: The RGCN model was trained on the data collected in 2022. Bot behavior evolves continuously, and accounts created after the training period may exhibit patterns not represented in the training distribution. The tweet-count-invariant embedding strategy and confidence calibration module partially mitigate this risk, but periodic retraining on updated data would be necessary for sustained accuracy.

Deprecated profile features: Six of the 20 profile features (profile_use_background_image, default_profile_background_color, default_profile_sidebar_fill_color, default_profile_sidebar_border_color, geo_enabled, profile_background_image_url) correspond to Twitter API v1.1 fields that are no longer returned by the current API. At inference time, these features default to static values and contribute no discriminative signal. The ablation study shows that removing all profile features causes only a 0.16% accuracy drop, suggesting the model is not critically dependent on these deprecated fields.

Ego-graph-scale disparity: The training graph contains 10,199 densely connected nodes with 1.7 million edges. Live inference constructs ego-graphs with typically 15-25 nodes and 20-50 edges. This scale disparity means the RGCN's neighborhood aggregation operates on substantially sparser input during inference than during training, which the confidence calibration module addresses but does not eliminate.

Limited live validation: The live inference results reported in this work cover a small number of manually tested accounts. Large-scale live validation across diverse account categories (bots, humans, celebrities, news organizations, political accounts) was not conducted due to scraper rate limitations.

Future work

Several promising directions for future research are envisioned:

- **Managed scraping infrastructure:** Migrating from cookie-based Scweet to a fully managed scraping service (e.g., Apify Actors) would eliminate authentication fragility and enable higher-throughput data acquisition.
- **Batch inference and botnet analysis:** Extending the system to accept CSV uploads of multiple usernames would enable large-scale botnet mapping and coordinated inauthentic behavior detection.
- **Temporal graph networks:** Incorporating temporal edge features (e.g., interaction timestamps, posting cadence patterns) through temporal graph network architectures could capture time-varying behavioral signatures.

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

- Adversarial robustness evaluation: Systematic evaluation against LLM-powered bots (e.g., GPT-4-driven accounts) and adversarial profile manipulation strategies would establish the model's resilience boundaries.
- Explainability integration: Augmenting predictions with GNNExplainer or attention-based explanations would provide actionable insights for content moderators, identifying which features and graph structural patterns drive classifications.
- Real-time graph expansion: Implementing a dynamic graph expansion mechanism that progressively discovers second-hop neighbors during inference could enhance classification confidence for accounts with sparse immediate neighborhoods.
- Cross-platform generalization: Evaluating the framework's transferability to other social media platforms (e.g., Mastodon, Bluesky, Reddit) with their distinct interaction modalities and network topologies would assess the generalizability of the multi-relational approach.

Conclusions

This paper has presented MTAB-Live, a production-grade framework for real-time social media bot detection that leverages multi-relational graph neural networks. By constructing dynamic ego-graphs with seven semantically distinct edge types and encoding nodes with 788-dimensional feature vectors combining normalized profile attributes with LaBSE sentence embeddings, our system captures the structural and behavioral signatures that distinguish automated accounts from genuine users. The RGCN architecture achieved $87.63\% \pm 1.20\%$ test accuracy and $90.45\% \pm 1.64\%$ bot recall across five independent evaluation seeds on the MTAB benchmark, outperforming GCN, GAT, and GraphSAGE baselines by significant margins. The system has been deployed as a full-stack web application with a React 19.2 frontend and dockerized FastAPI backend, demonstrating the feasibility of graph neural network deployment in production environments. Key engineering contributions include an SSE streaming pipeline for real-time progress communication, a rate-limit-resilient scraper architecture with retry-backoff and graceful degradation to noise-vector substitution, a serverless Redis caching layer for instant repeat-query responses, a post-hoc confidence calibration module for out-of-distribution inputs, and a rigorously validated feature engineering pipeline that employs a mean-pool, L2-normalize, and fixed-norm rescaling strategy to produce tweet-count-invariant embeddings suitable for variable scrape yields in production.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Arihant Jain, Pratham Jaulkar, Aayush Agrawal, Frank Garvin

Acquisition, analysis, or interpretation of data: Arihant Jain, Pratham Jaulkar, Aayush Agrawal, Frank Garvin, Rajesh Khobragade

Drafting of the manuscript: Arihant Jain, Pratham Jaulkar, Aayush Agrawal, Frank Garvin

Critical review of the manuscript for important intellectual content: Arihant Jain, Pratham Jaulkar, Aayush Agrawal, Frank Garvin, Rajesh Khobragade

Supervision: Rajesh Khobragade

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

Data Availability Statements

The program code and full-stack implementation supporting this study are openly available in GitHub at <https://github.com/Arihant0008/MGTAB>. The MTAB benchmark dataset analyzed during this study is openly available in GitHub at <https://github.com/GraphDetec/MGTAB>. Additionally, the live real-time bot detection web application discussed in this study is publicly accessible at <https://www.mgtab.me>.

References

1. Ke Q, Ahn YY, Sugimoto CR: [A systematic identification and analysis of scientists on Twitter](#). PLOS One. 2017, 12:1-17. [10.1371/journal.pone.0175368](#)
2. Cresci S: [A decade of social bot detection](#). Communications of the ACM. 2020, 63:72-83. [10.1145/3409116](#)
3. Ferrara E, Varol O, Davis C, Menczer F, Flammini A: [The rise of social bots](#). Communications of the ACM. 2016, 59:96-104. [10.1145/2818717](#)
4. Kwon S, Jang B: [A comprehensive survey of fake text detection on misinformation and LM-generated texts](#). IEEE Access: Practical Innovations, Open Solutions. 2025, 13:25301-25324. [10.1109/access.2025.3538805](#)
5. Cresci S, Di Pietro R, Petrocchi M, Spognardi A, Tesconi M: [The paradigm-shift of social spambots: evidence, theories, and tools for the arms race](#). WWW '17 Companion: Proceedings of the 26th International Conference on World Wide Web Companion. 2017, 963-972. [10.1145/3041021.3055135](#)
6. Shao C, Ciampaglia GL, Varol O, Yang KC, Flammini A, Menczer F: [The spread of low-credibility content by social bots](#). Nature Communications. 2018, 9:1-9. [10.1038/s41467-018-06930-7](#)
7. Alhosseini F, Bin Tareaf A, Najafi P, Meinel C: [Detect me if you can: spam bot detection using inductive representation learning](#). WWW '19: Companion Proceedings of the 2019 World Wide Web Conference. 2019, 148-153. [10.1145/3308560.3316504](#)
8. Varol O, Ferrara E, Davis C, Menczer F, Flammini A: [Online human-bot interactions: detection, estimation, and characterization](#). Proceedings of the International AAAI Conference on Weblogs and Social Media. 2017, 11:280-289. [10.1609/icwsm.v11i1.14871](#)
9. Cresci S, Di Pietro R, Petrocchi M, Spognardi A, Tesconi M: [DNA-inspired online behavioral modeling and its application to spambot detection](#). IEEE Intelligent Systems. 2016, 31:58-64. [10.1109/mis.2016.29](#)
10. Yang KC, Varol O, Hui PM, Menczer F: [Scalable and generalizable social bot detection through data selection](#). Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34:1096-1103. [10.1609/aaai.v34i01.5460](#)
11. Feng S, Wan H, Wang N, Luo M: [BotRGCN: Twitter bot detection with relational graph convolutional networks](#). arXiv. 2021, 1-4. [10.48550/arXiv.2106.13092](#)
12. Feng S, Wan H, Wang N, Li J, Luo M: [TwiBot-20: a comprehensive Twitter bot detection benchmark](#). CIKM '21: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2021, 4485-4494. [10.1145/3459637.3482019](#)
13. Feng S, Tan Z, Wan H, et al.: [M: TwiBot-22: towards graph-based Twitter bot detection](#). arXiv. 2022, 1-25.
14. Kipf TN, Welling M: [Semi-supervised classification with graph convolutional networks](#). Proceeding of ICLR. 2017, 1-14.
15. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y: [Graph attention networks](#). Proceedings of ICLR. 2018, 1-12.
16. Hamilton WL, Ying R, Leskovec J: [Inductive representation learning on large graphs](#). Advances in Neural Information Processing Systems. 2017, 30:1024-1034.
17. Schlichtkrull M, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M : [Modeling relational data with graph convolutional networks](#). Lecture Notes in Computer Science. 2018, 10843:593-607. [10.1007/978-3-319-93417-4_38](#)
18. Shi S, Qiao K, Chen, et al.: [MGTab: a multi-relational graph-based Twitter account detection benchmark](#). arXiv. 2023, 1-14. [10.48550/arXiv.2301.01123](#)

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>

19. Feng F, Yang Y, Cer D, Arivazhagan N, Wang W: [Language-agnostic: BERT sentence embedding](#). Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2022, 878-891. [10.18653/v1/2022.acl-long.62](#)
20. Kingma DP, Ba J: [Adam: a method for stochastic optimization](#). 3rd International Conference for Learning Representations. 2015, 1-15.
21. Rossi E, Chamberlain B, Frasca F, Eynard D, Monti F, Bronstein M: [Temporal graph networks for deep learning on dynamic graphs](#). arXiv. 2020, 1-16. [10.48550/arXiv.2006.10637](#)

How to cite this article:

Jain A, Jaulkar P, Agrawal A, et al. (July 01, 2026) MTAB-Live: A Production-Ready Multi-Relational Graph-Based Framework for Real-Time Twitter Bot Detection Using Relational Graph Convolutional Networks. Cureus J Comput Sci 3 : es44389-026-00148-1. DOI <https://doi.org/10.7759/s44389-026-00148-1>