Bot-or-Not:

Temporal Graph-Based Bot Detection on Twitter

Team 03

Komal Chandiramani
Madhav Walia
Tanya Warrier

Image source: Freepik

Motivation

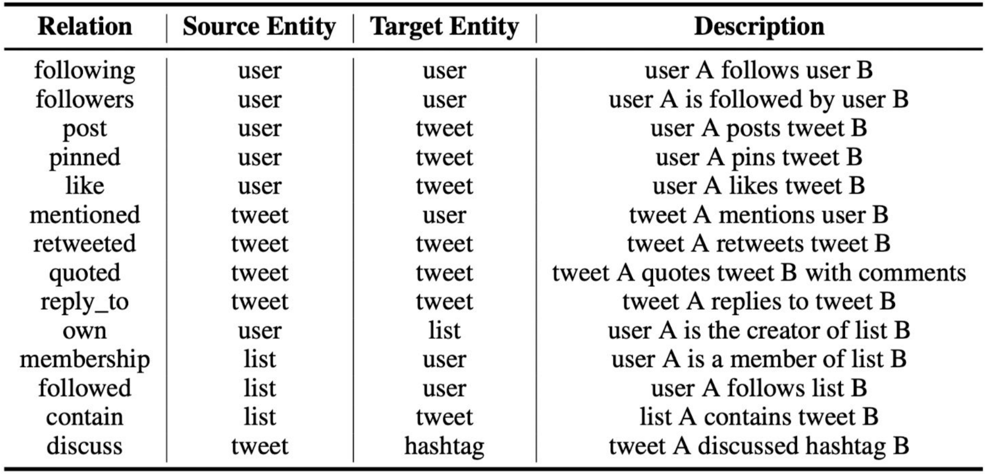
- Our task is to **classify user nodes** as humans or bots
- Bot behaviour is **dynamic**, not static
- Most graph-based bot detectors ignore time
- Twitter graph is inherently **temporal**

Project goal

Construct a discrete-time dynamic graph (**DTDG**) from TwiBot-22 and apply RGCN over temporal snapshots, to capture evolving relational patterns and improve robustness to distribution shift in bot detection.

Twibot 22

- Largest Bot Detection benchmark dataset to date
- Contains **1 million users** and **88 million tweets** collected from Jan 2022 - Feb 2022
- 4 entity types: User, Tweet, Lists and Hashtags
- 14 relation types:



Relation	Source Entity	Target Entity	Description
following	user	user	user A follows user B
followers	user	user	user A is followed by user B
post	user	tweet	user A posts tweet B
pinned	user	tweet	user A pins tweet B
like	user	tweet	user A likes tweet B
mentioned	tweet	user	tweet A mentions user B
retweeted	tweet	tweet	tweet A retweets tweet B
quoted	tweet	tweet	tweet A quotes tweet B with comments
reply_to	tweet	tweet	tweet A replies to tweet B
own	user	list	user A is the creator of list B
membership	list	user	user A is a member of list B
followed	list	user	user A follows list B
contain	list	tweet	list A contains tweet B
discuss	tweet	hashtag	tweet A discussed hashtag B

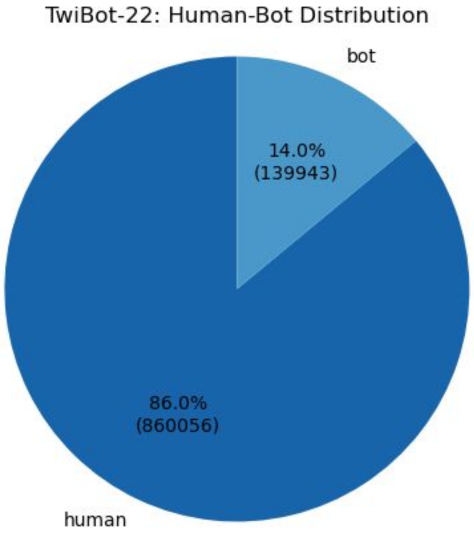
# Nodes	21,359
#Edges	795,397
# Entity Types	1 (user)
# Relation Types	2 (*follower* and *following*)

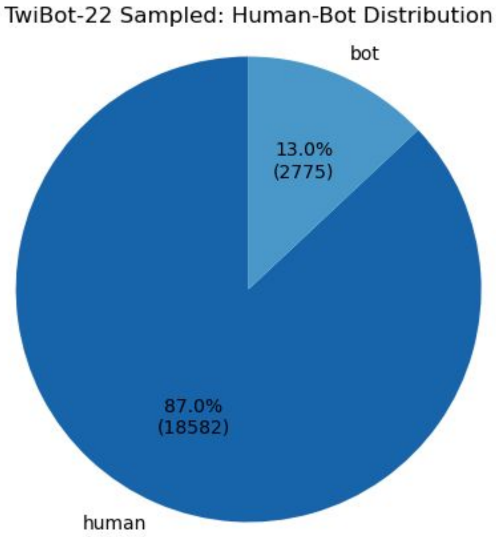
Summary Statistics of the Subsampled Graph

Sampling

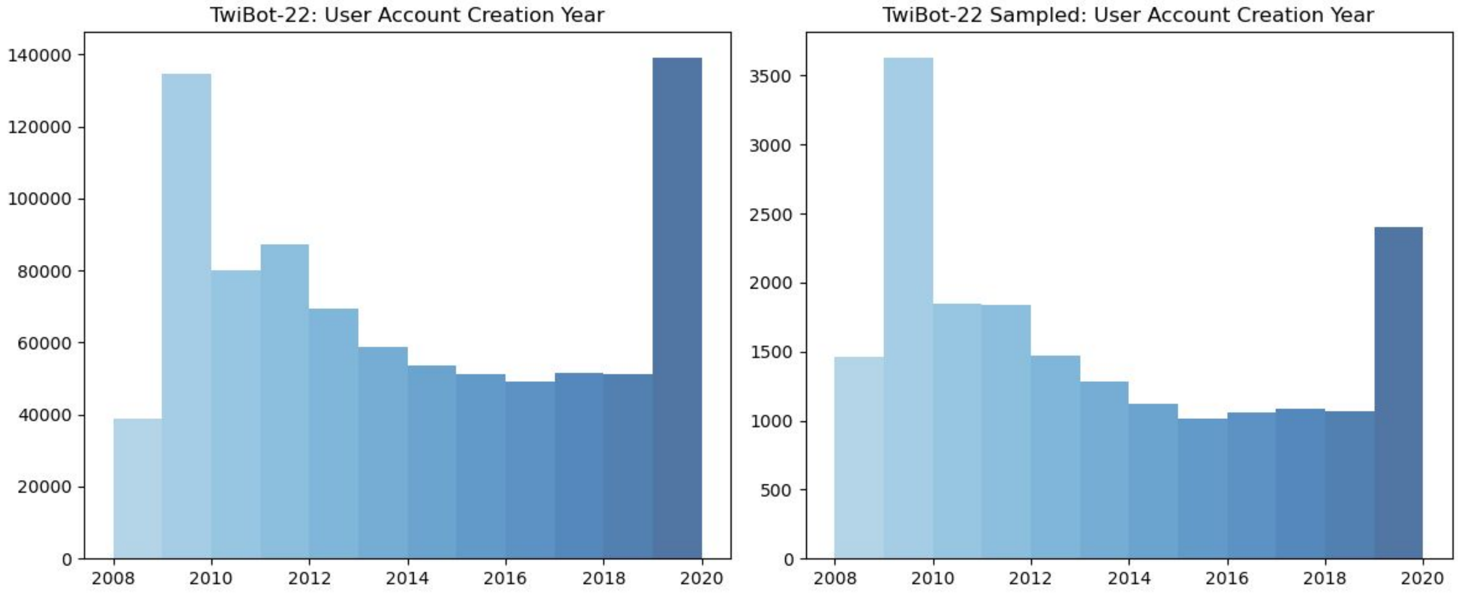
- Dataset subsampled in 3 stages:
  - Stage 1: Seed Selection
  - Stage 2: Graph Expansion
  - Stage 3: Entity Extraction

The sampling pipeline ensures that we maintain:
Human-Bot label distribution

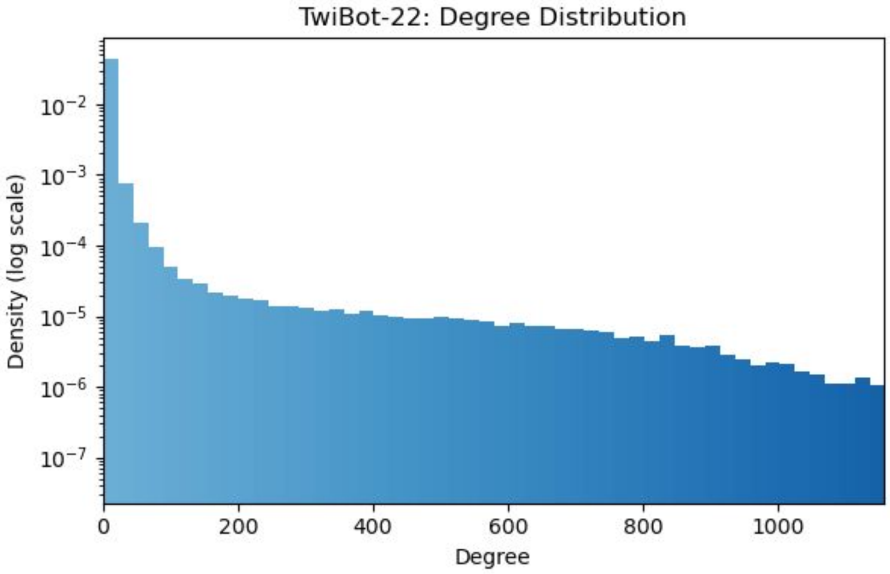


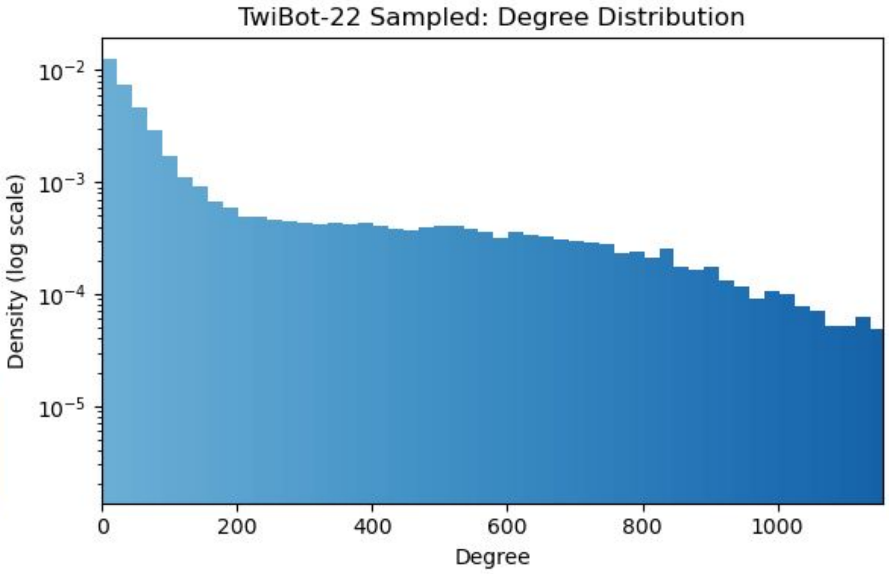


The sampling pipeline ensures that we maintain:
User Account Creation Temporal Distribution

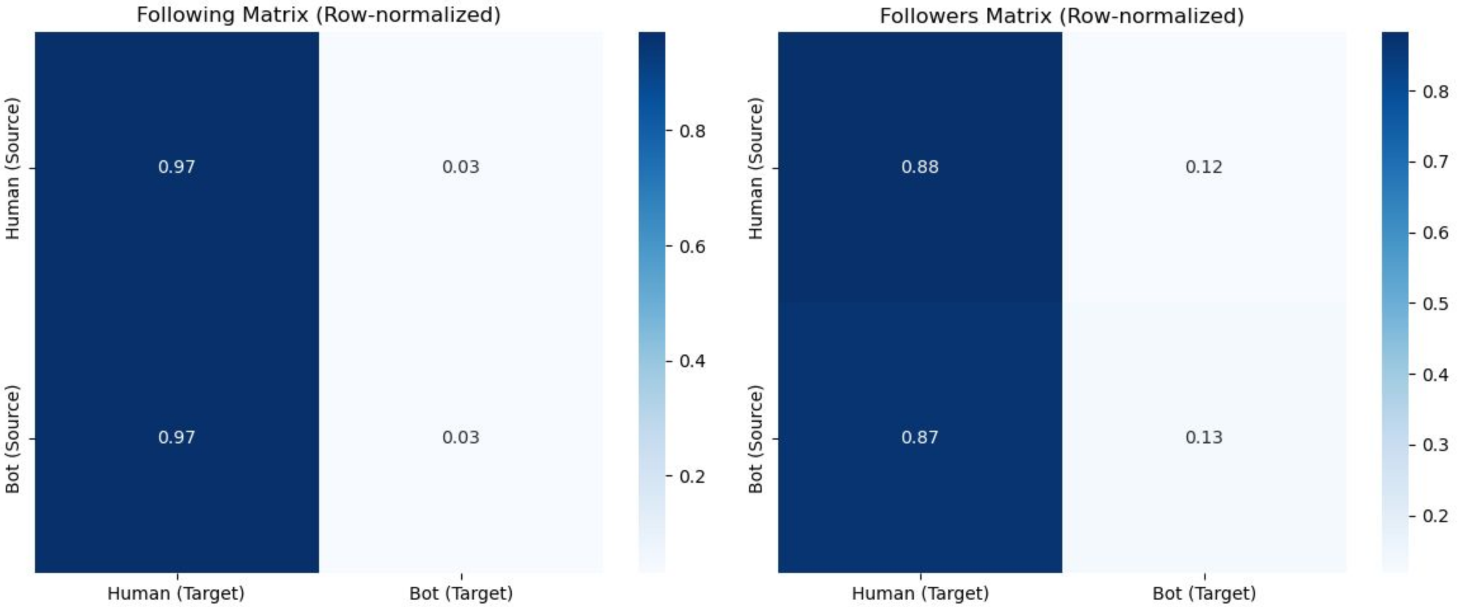


The sampling pipeline ensures that we maintain:
Degree Distribution

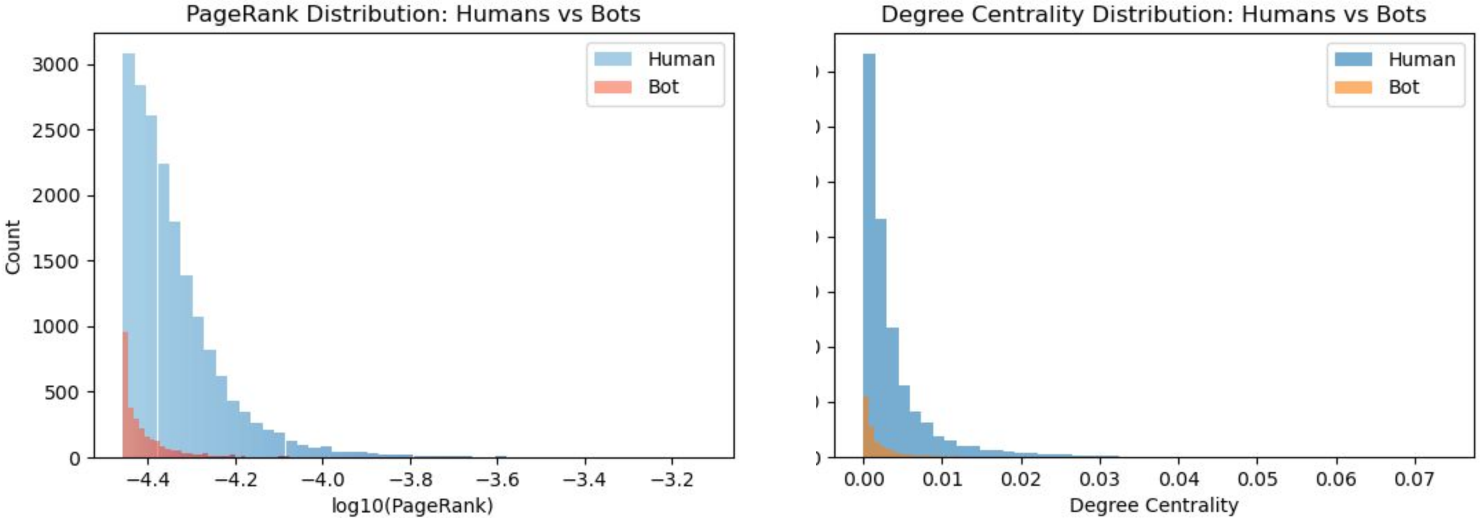




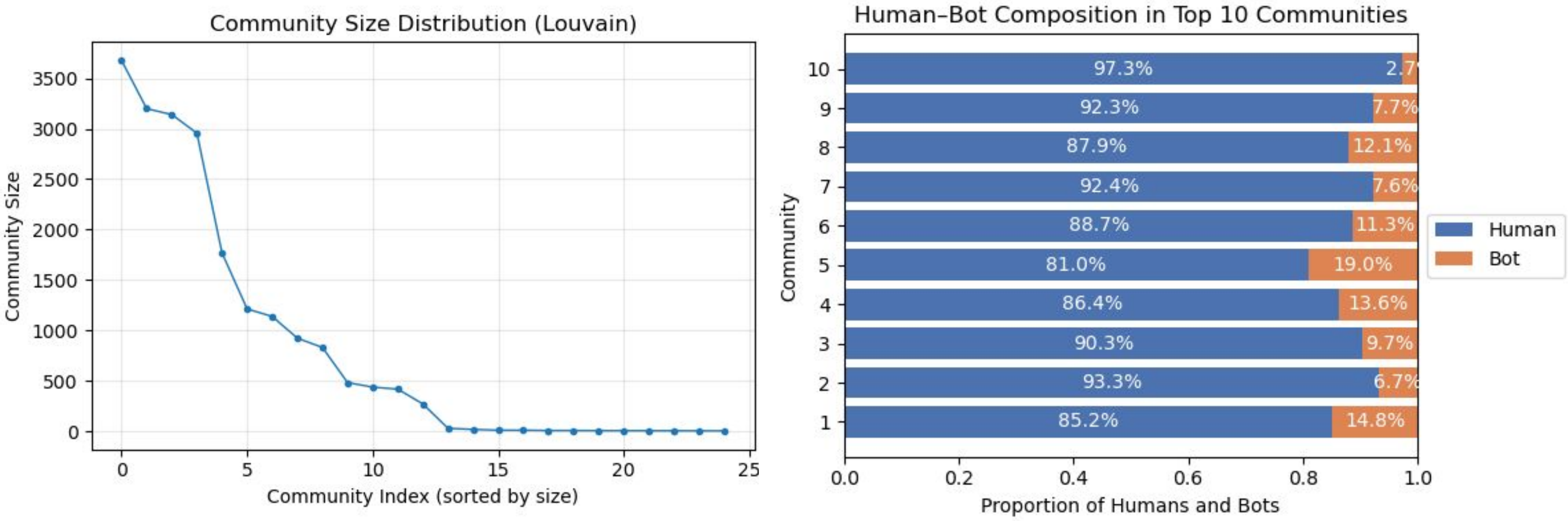
Exploratory Data Analysis



Human - Bot Interaction Confusion Matrix



Centrality Measures Comparison between Humans and Bots



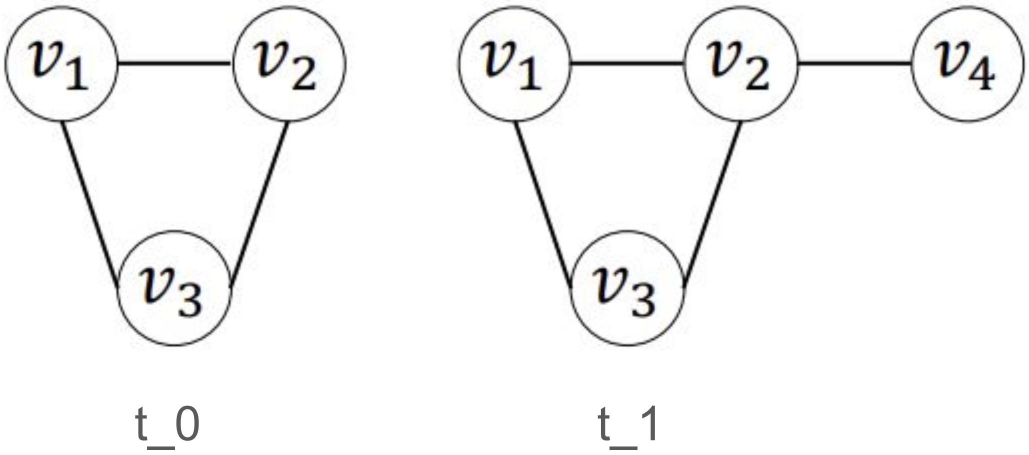
Community Size and Human-Bot Distribution

Discrete Time Dynamic Graph

- There are Two Types of Dynamic Graphs:

Discrete Time Dynamic Graph

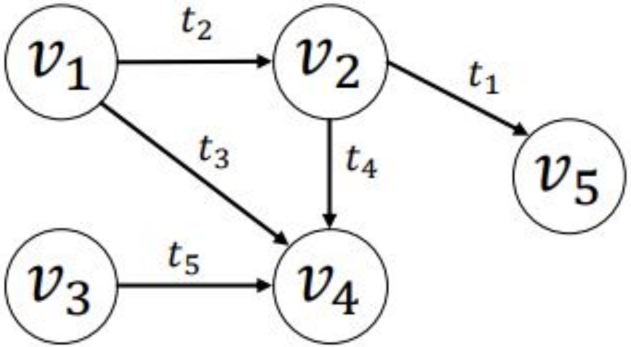
- There are Two Types of Dynamic Graphs:
  - **Discrete Time Dynamic Graph (DTDG):** These are sequence of snapshots (G_0, G_1, ..., G□) at fixed time intervals 0 to t.



Source: Kazemi et al. (2020), JMLR, Fig. 1

Discrete Time Dynamic Graph

- There are Two Types of Dynamic Graphs:
  - **Discrete Time Dynamic Graph (DTDG):** These are sequence of snapshots (G_0, G_1, ..., G□) at fixed time intervals 0 to t.
  - **Continuous Time Dynamic Graph (CTDG):** These have a tuple of (event type, event, timestamp) relations and capture edge/node additions, deletions, splitting, merging.



Source: Kazemi et al. (2020), JMLR, Fig. 1

Discrete Time Dynamic Graph

- There are Two Types of Dynamic Graphs:
  - ○ **Discrete Time Dynamic Graph (DTDG):** These are sequence of snapshots (G_0, G_1, ..., G□) at fixed time intervals 0 to t.
  - ○ **Continuous Time Dynamic Graph (CTDG):** These have (event type, event, timestamp) relations and captures edge/node additions, deletions, splitting, merging.
- While CTDGs tend to be expressive, we chose DTDGs due to the following considerations:

Discrete Time Dynamic Graph

- There are Two Types of Dynamic Graphs:
  - **Discrete Time Dynamic Graph (DTDG):** These are sequence of snapshots (G_0, G_1, ..., G□) at fixed time intervals 0 to t.
  - **Continuous Time Dynamic Graph (CTDG):** These have (event type, event, timestamp) relations and captures edge/node additions, deletions, splitting, merging.
- While CTDGs tend to be expressive, we chose DTDGs due to the following considerations:
  - No edge timestamps

Discrete Time Dynamic Graph

- There are Two Types of Dynamic Graphs:
  - **Discrete Time Dynamic Graph (DTDG):** These are sequence of snapshots (G_0, G_1, ..., G□) at fixed time intervals 0 to t.
  - **Continuous Time Dynamic Graph (CTDG):** These have (event type, event, timestamp) relations and captures edge/node additions, deletions, splitting, merging.
- While CTDGs tend to be expressive, we chose DTDGs due to the following considerations:
  - No edge timestamps
  - No edge/node deletion information

Discrete Time Dynamic Graph

- There are Two Types of Dynamic Graphs:
  - **Discrete Time Dynamic Graph (DTDG):** These are sequence of snapshots (G_0, G_1, ..., G□) at fixed time intervals 0 to t.
  - **Continuous Time Dynamic Graph (CTDG):** These have (event type, event, timestamp) relations and captures edge/node additions, deletions, splitting, merging.
- While CTDGs tend to be expressive, we chose DTDGs due to the following considerations:
  - No edge timestamps
  - No edge/node deletion information
- Our temporalization process is as follows:

Discrete Time Dynamic Graph

- There are Two Types of Dynamic Graphs:
  - **Discrete Time Dynamic Graph (DTDG):** These are sequence of snapshots (G_0, G_1, ..., G□) at fixed time intervals 0 to t.
  - **Continuous Time Dynamic Graph (CTDG):** These have (event type, event, timestamp) relations and captures edge/node additions, deletions, splitting, merging.
- While CTDGs tend to be expressive, we chose DTDGs due to the following considerations:
  - No edge timestamps
  - No edge/node deletion information
- Our temporalization process is as follows:
  - Extract account creation times for users and tweets

Discrete Time Dynamic Graph

- There are Two Types of Dynamic Graphs:
  - **Discrete Time Dynamic Graph (DTDG):** These are sequence of snapshots (G_0, G_1, ..., G□) at fixed time intervals 0 to t.
  - **Continuous Time Dynamic Graph (CTDG):** These have (event type, event, timestamp) relations and captures edge/node additions, deletions, splitting, merging.
- While CTDGs tend to be expressive, we chose DTDGs due to the following considerations:
  - No edge timestamps
  - No edge/node deletion information
- Our temporalization process is as follows:
  - Extract account creation times for users and tweets
  - Infer edge activation time as the later creation timestamp of its two endpoint

Discrete Time Dynamic Graph

- There are Two Types of Dynamic Graphs:
  - **Discrete Time Dynamic Graph (DTDG):** These are sequence of snapshots (G_0, G_1, ..., G□) at fixed time intervals 0 to t.
  - **Continuous Time Dynamic Graph (CTDG):** These have (event type, event, timestamp) relations and captures edge/node additions, deletions, splitting, merging.
- While CTDGs tend to be expressive, we chose DTDGs due to the following considerations:
  - No edge timestamps
  - No edge/node deletion information
- Our temporalization process is as follows:
  - Extract account creation times for users and tweets
  - Infer edge activation time as the later creation timestamp of its two endpoints
  - Define weekly time grid from earliest to latest timestamp and build cumulative snapshots: users → tweets → edges progressively added to each week

Feature Engineering

- Bot detection depends heavily on user behaviour

Feature Engineering

- Bot detection depends heavily on user behaviour

- Need expressive node embeddings

Feature Engineering

- Bot detection depends heavily on user behaviour
- Need expressive node embeddings
- Inspired by the **BotRGCN** work, we build similar lightweight features.

Feng, S., Wan, H., Wang, N., & Luo, M. (2021, November). BotRGCN: Twitter bot detection with relational graph convolutional networks. In Proceedings of the 2021 IEEE/ACM international conference on advances in social networks analysis and mining (pp. 236-239).

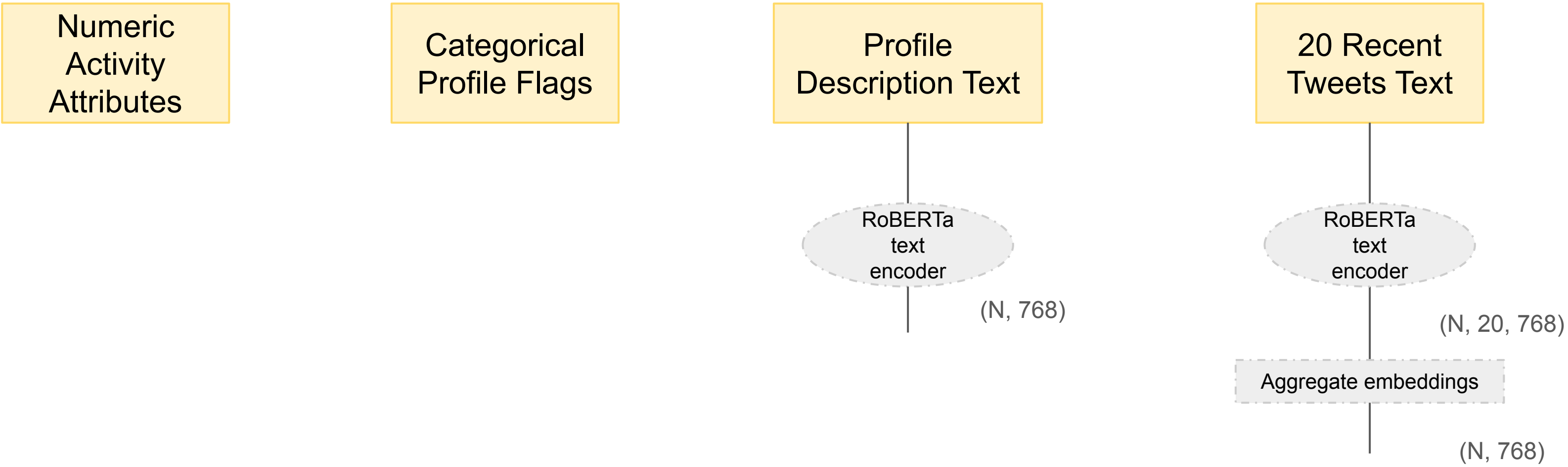
Static Graph Features

Static Graph Features

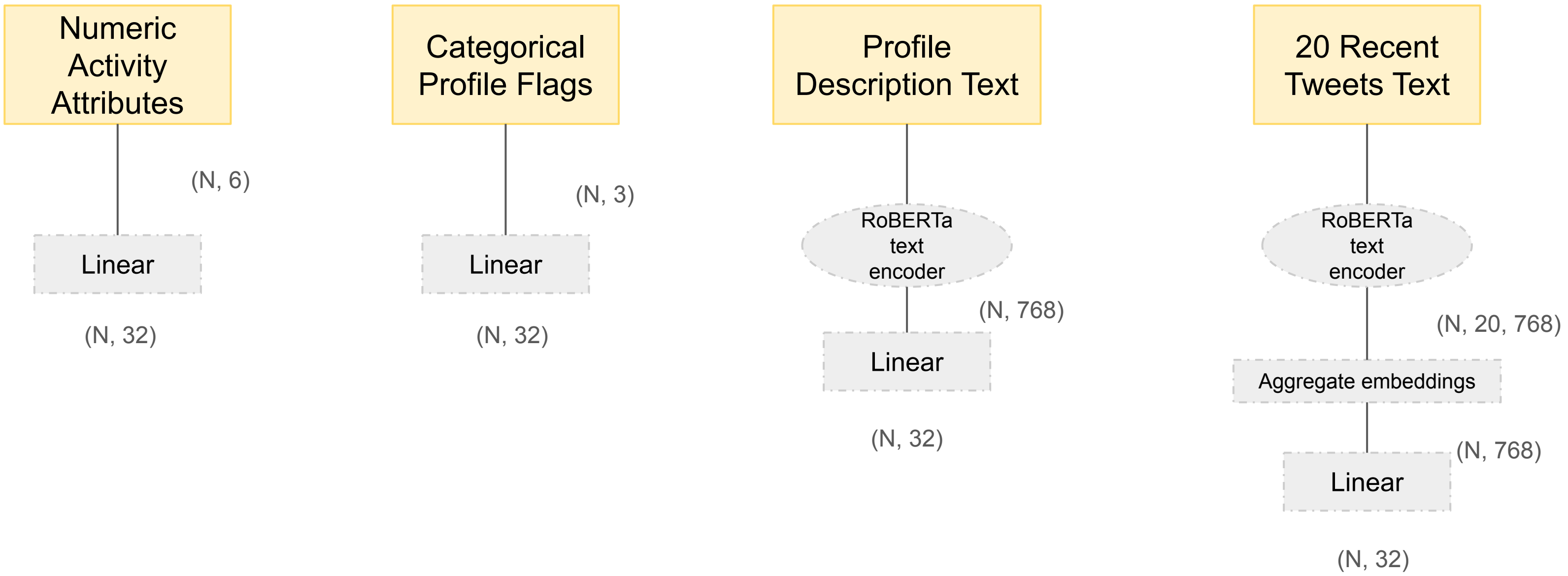
Numeric Activity Attributes

Categorical Profile Flags

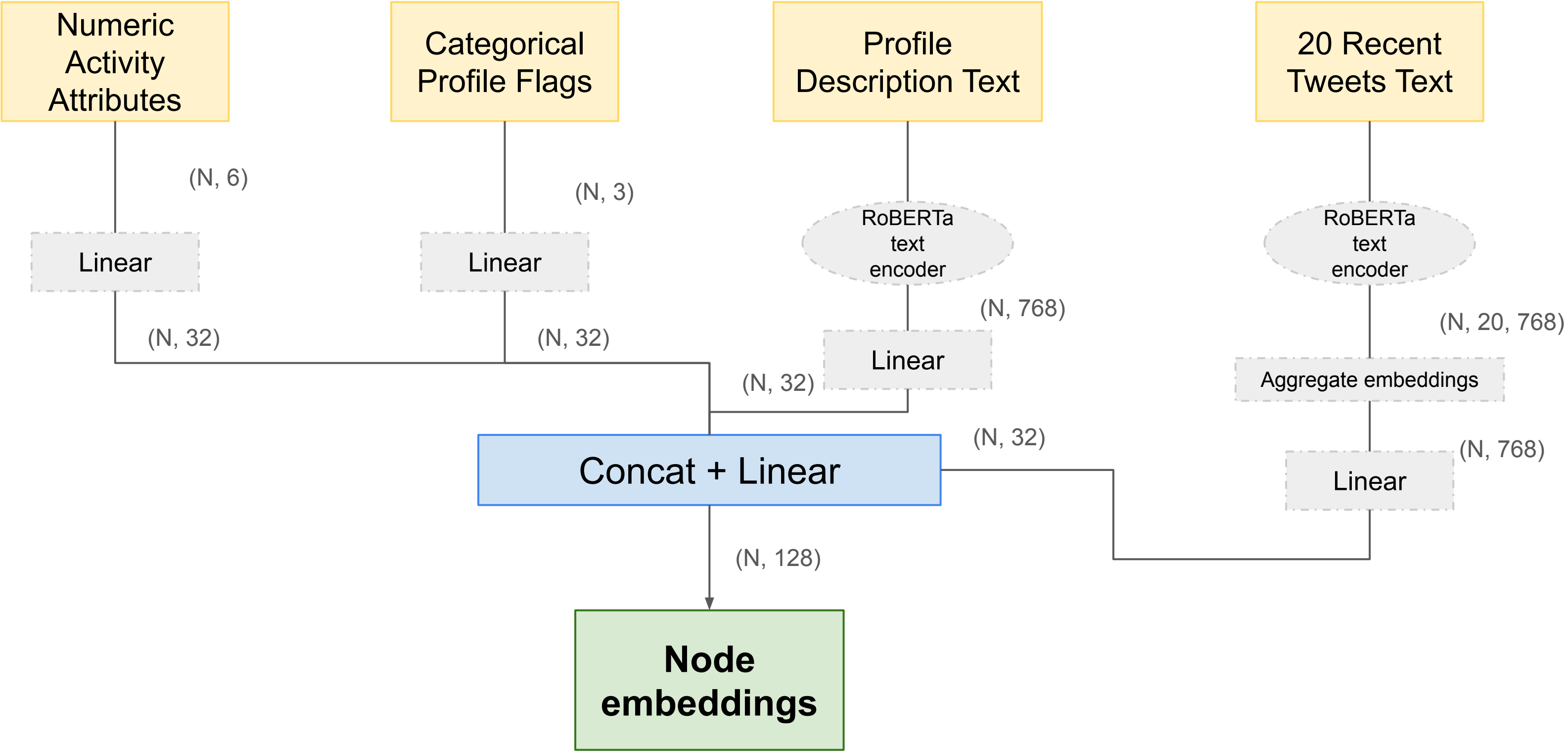
Static Graph Features



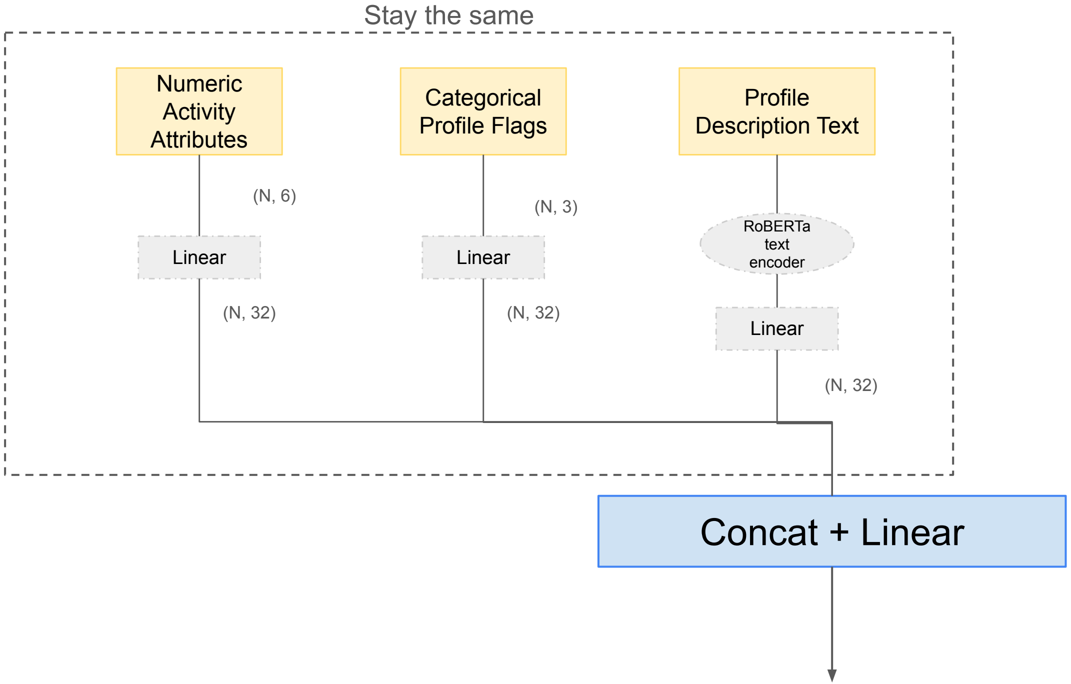
Static Graph Features



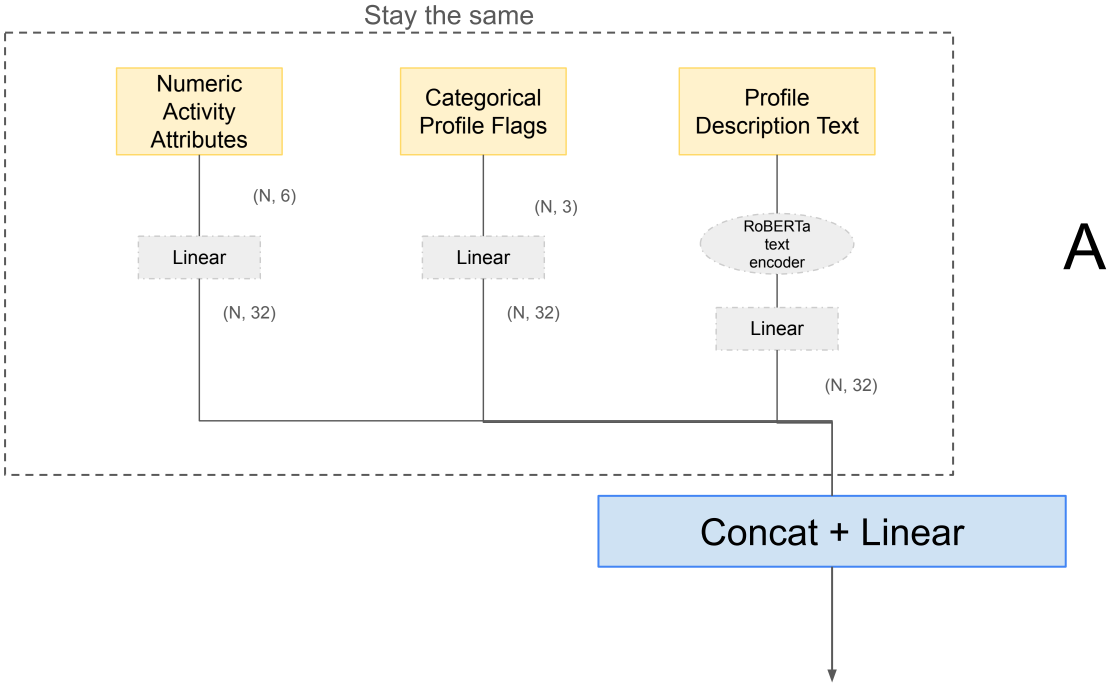
Static Graph Features



DTDG Features

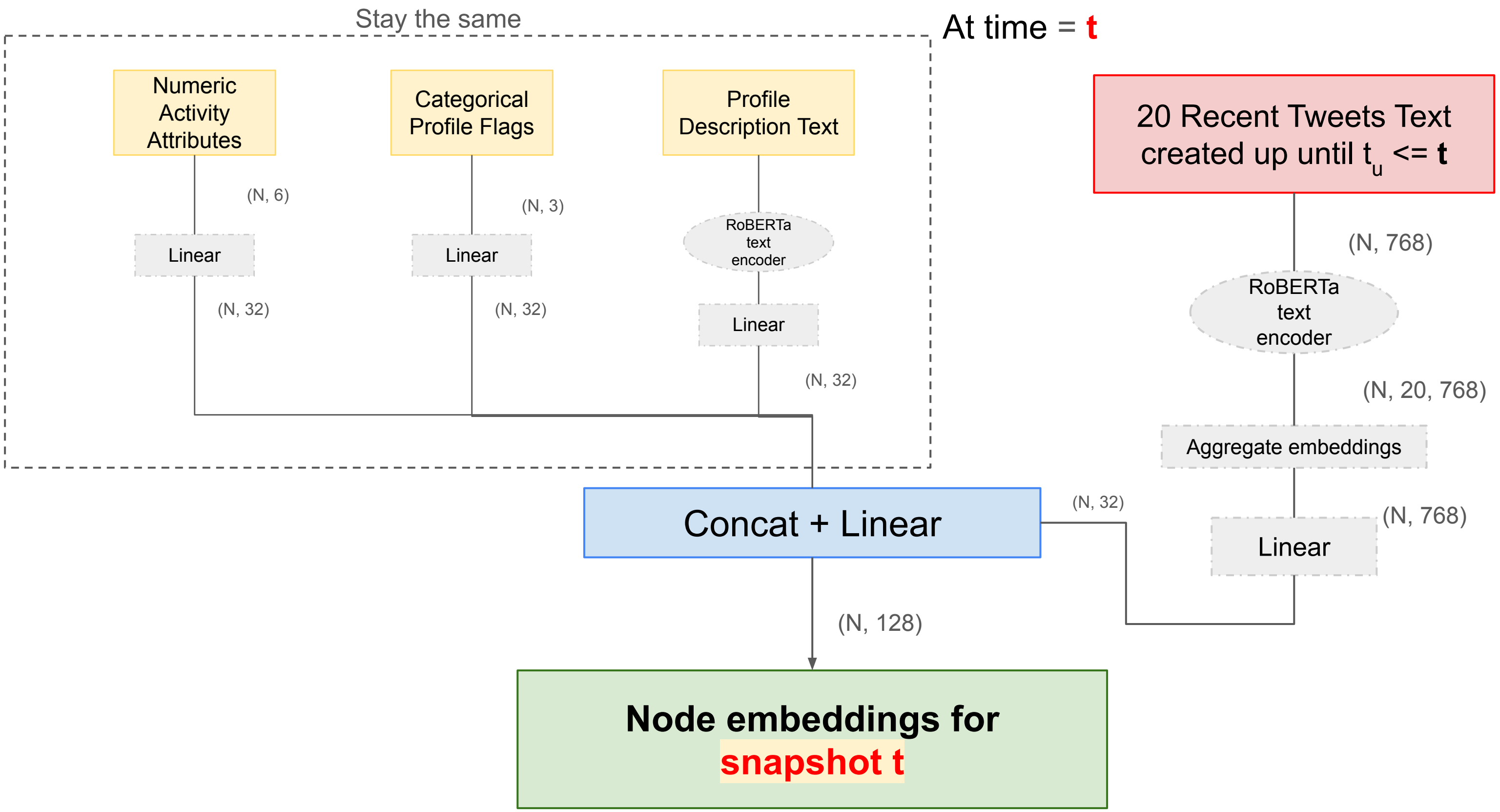


DTDG Features



At time = t

DTDG Features



Stay the same

At time = **t**

Numeric Activity Attributes

(N, 6)

Linear

(N, 32)

Categorical Profile Flags

(N, 3)

Linear

(N, 32)

Profile Description Text

RoBERTa text encoder

Linear

(N, 32)

20 Recent Tweets Text created up until t_u <= **t**

(N, 768)

RoBERTa text encoder

(N, 20, 768)

Aggregate embeddings

(N, 768)

Linear

(N, 32)

Concat + Linear

(N, 128)

Node embeddings for snapshot t

BotRGCN

- Graph edges encode two **asymmetric** relations:
  - user follows another (*following*)
  - is followed by another (*follower*)

BotRGCN

- Graph edges encode two **asymmetric** relations:
  - user follows another (*following*)
  - is followed by another (*follower*)
- Vanilla GCN treats edges as homogeneous, loses **social network semantics**.

BotRGCN

- Graph edges encode two **asymmetric** relations:
    - user follows another (*following*)
    - is followed by another (*follower*)
- Vanilla GCN treats edges as homogeneous, loses **social network semantics**.
- BotRGCN assigns relation-specific transformations for the 2 relations

BotRGCN

- Graph edges encode two **asymmetric** relations:
  - user follows another (*following*)
  - is followed by another (*follower*)
- Vanilla GCN treats edges as homogeneous, loses **social network semantics**.
- BotRGCN assigns relation-specific transformations for the 2 relations
- Since each relation has its **own weight matrix**
  - the model can learn patterns like ‘follows many people but followed by few’

BotRGCN

Architecture

1. 2-layer R-GCN
   - 128 → 128 (LeakyReLU, dropout)
   - 128 → 128 (LeakyReLU)
2. MLP classifier
   - 128 → 128 → 2 (logits)
3. Trained on static graph with node embeddings.

Static embeddings ignore evolution of user behaviour.

Temporal BotRGCN: EvoRGCN

- **Principle:** Instead of having a sequential model measure the change in node embeddings, we evolve the weight matrices themselves through a matrix-valued GRU.[1]

[1] Pareja et al. (2020), EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs.

Temporal BotRGCN: EvoRGCN

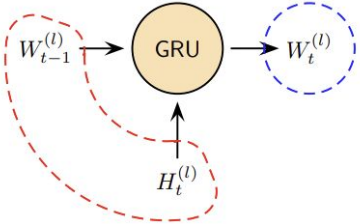
- **Principle:** Instead of of having a sequential model measure the change in node embeddings, we evolve the weight matrices themselves through a matrix-valued GRU.[1]
- **Architecture:**

[1] Pareja et al. (2020), EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs.

Temporal BotRGCN: EvoRGCN

- **Principle:** Instead of of having a sequential model measure the change in node embeddings, we evolve the weight matrices themselves through a matrix-valued GRU.[1]

- **Architecture:**
  - **Step 1 - Weight Evolution**:
    - **Input:** Previous weights W^{l}_{t-1} (red dashed) + Current node embeddings H^{l}_{t} (red dashed)
    - **Process:** MatGRU takes both inputs
    - **Output:** Evolved weights W^{l}_{t} (blue dashed)

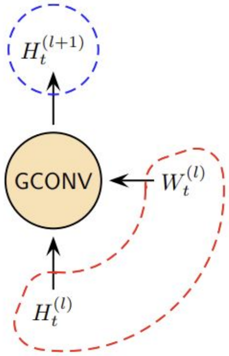


[1] Pareja et al. (2020), EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs.

Temporal BotRGCN: EvoRGCN

- **Principle:** Instead of of having a sequential model measure the change in node embeddings, we evolve the weight matrices themselves through a matrix-valued GRU.[1]

- **Architecture:**
  - **Step 2 - Relational Graph Convolution**:
    - **Input:** Current node embeddings H^l_t + Evolved weights W^l_t
    - **Process:** RGCN operation
    - **Output:** Next layer embeddings H^l_{t+1} (blue dashed)



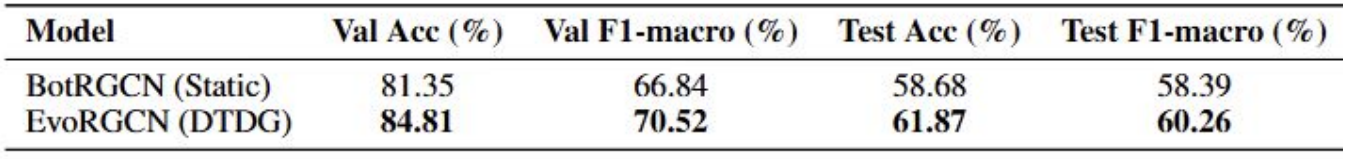
[1] Pareja et al. (2020), EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs.

Results

- Model Performance on Sampled TwiBot-22 Graph:

Results

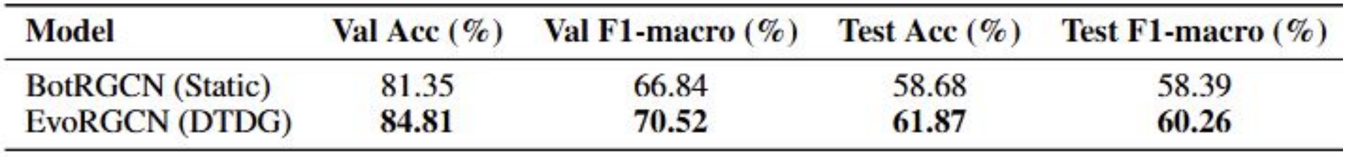
- Model Performance on Sampled TwiBot-22 Graph:



Model	Val Acc (%)	Val F1-macro (%)	Test Acc (%)	Test F1-macro (%)
BotRGCN (Static)	81.35	66.84	58.68	58.39
EvoRGCN (DTDG)	**84.81**	**70.52**	**61.87**	**60.26**

Results

- Model Performance on Sampled TwiBot-22 Graph:

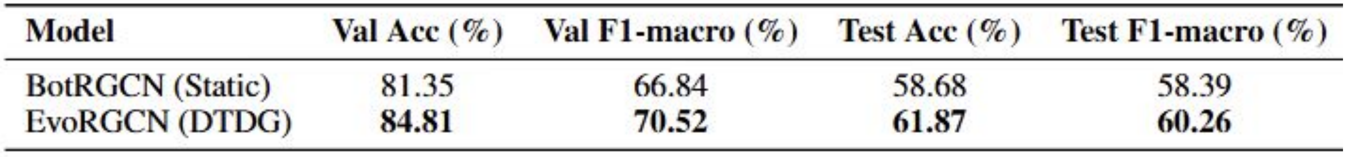


Model	Val Acc (%)	Val F1-macro (%)	Test Acc (%)	Test F1-macro (%)
BotRGCN (Static)	81.35	66.84	58.68	58.39
EvoRGCN (DTDG)	**84.81**	**70.52**	**61.87**	**60.26**

- We give more weightage to F1 score over accuracy due to the severe class imbalance: predicting "all human" gives ~87% accuracy. Improvement in F1 shows EvoRGCN better detects minority class (bots)

Results

- Model Performance on Sampled TwiBot-22 Graph:

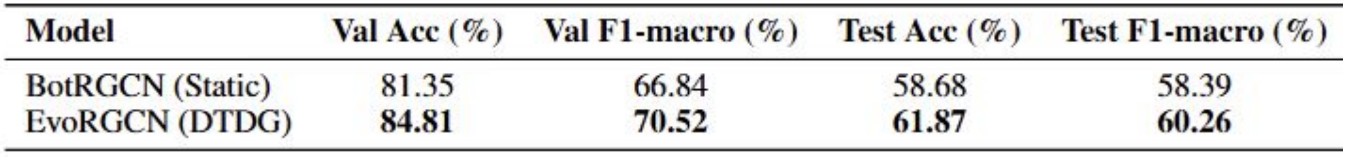


Model	Val Acc (%)	Val F1-macro (%)	Test Acc (%)	Test F1-macro (%)
BotRGCN (Static)	81.35	66.84	58.68	58.39
EvoRGCN (DTDG)	**84.81**	**70.52**	**61.87**	**60.26**

- We give more weightage to F1 score over accuracy due to the severe class imbalance: predicting "all human" gives ~87% accuracy. Improvement in F1 shows EvoRGCN better detects minority class (bots)
- Key advantages of temporal modeling:

Results

- Model Performance on Sampled TwiBot-22 Graph:



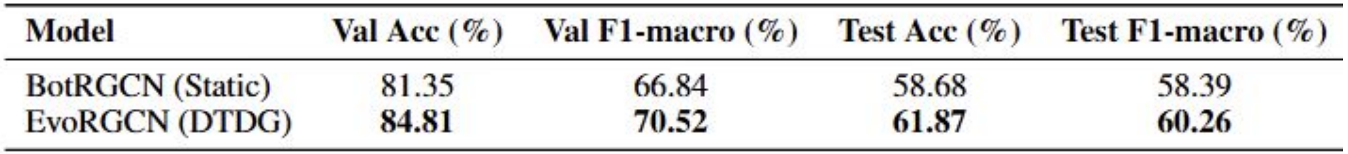
Model	Val Acc (%)	Val F1-macro (%)	Test Acc (%)	Test F1-macro (%)
BotRGCN (Static)	81.35	66.84	58.68	58.39
EvoRGCN (DTDG)	**84.81**	**70.52**	**61.87**	**60.26**

- We give more weightage to F1 score over accuracy due to the severe class imbalance: predicting "all human" gives ~87% accuracy. Improvement in F1 shows EvoRGCN better detects minority class (bots)

- Key advantages of temporal modeling:
  - Captures evolving degree patterns and recent tweet behavior

Results

- Model Performance on Sampled TwiBot-22 Graph:



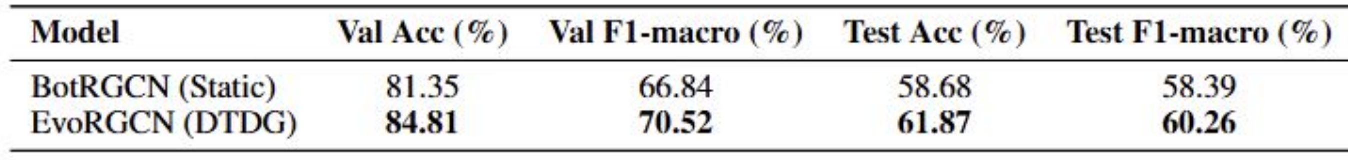
Model	Val Acc (%)	Val F1-macro (%)	Test Acc (%)	Test F1-macro (%)
BotRGCN (Static)	81.35	66.84	58.68	58.39
EvoRGCN (DTDG)	**84.81**	**70.52**	**61.87**	**60.26**

- We give more weightage to F1 score over accuracy due to the severe class imbalance: predicting "all human" gives ~87% accuracy. Improvement in F1 shows EvoRGCN better detects minority class (bots)

- Key advantages of temporal modeling:
  - Captures evolving degree patterns and recent tweet behavior
  - Models behavioral drift and evolving bot strategies (static models cannot)

Results

- Model Performance on Sampled TwiBot-22 Graph:



Model	**Val Acc (%)**	**Val F1-macro (%)**	**Test Acc (%)**	**Test F1-macro (%)**
BotRGCN (Static)	81.35	66.84	58.68	58.39
EvoRGCN (DTDG)	**84.81**	**70.52**	**61.87**	**60.26**

- We give more weightage to F1 score over accuracy due to the severe class imbalance: predicting "all human" gives ~87% accuracy. Improvement in F1 shows EvoRGCN better detects minority class (bots)

- Key advantages of temporal modeling:
  - Captures evolving degree patterns and recent tweet behavior
  - Models behavioral drift and evolving bot strategies (static models cannot)
  - Better handles distribution shift to future users

Future Work

- Increase the Graph to more heterogeneous nodes (Tweet Nodes, etc.)

Future Work

- Increase the Graph to more heterogeneous nodes (Tweet Nodes, etc.)
- Increase various homophilic and heterophilic edge types (for example: post, retweet, like, etc.)

Future Work

- Increase the Graph to more heterogeneous nodes (Tweet Nodes, etc.)
- Increase various homophilic and heterophilic edge types (for example: post, retweet, like, etc.)
- Add more features like degree, centrality, page rank etc per snapshot

Future Work

- Increase the Graph to more heterogeneous nodes (Tweet Nodes, etc.)
- Increase various homophilic and heterophilic edge types (for example: post, retweet, like, etc.)
- Add more features like degree, centrality, page rank etc per snapshot
- Deploy our model to a sophisticated bot-detection pipeline with simple heuristics before applying our detector.