Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Section A.6
   (d) Did you describe the limitations of your work? [Yes] Section A.6
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please refer to the TwiBot-22 GitHub repository listed in Section A.6.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please refer to the TwiBot-22 GitHub repository listed in Section A.6.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Section B.6
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Section A.6
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide the TwiBot-22 dataset as a new asset and provide URLs in Section A.6.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We strictly follow the original license of existing datasets and rules in the Twitter Developer Agreement and Policy.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Our dataset does not contain the information of private and protected users. Since TwiBot-22 aims to facilitate bot detection research and certain bots are designed to be offensive, there might be offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Table 4: Entities in the TwiBot-22 heterogeneous graph.

Entity Name	Description	Main Metadata
User	Users are the most important entity on Twittersphere.	created at, description, entities location, name, profile image url protected, url, username verified, withheld, followers count following count, tweet count, listed count
Tweet	Users post tweets to share their thoughts and interact with other users.	attachments, context annotations, entities created at, geo, lang, possibly sensitive referenced tweets, reply settings source, text, withheld, retweet count reply count, like count, quote count
List	A list is curated feeds from selected users that allow you to listen to relevant discussions or influencers.	private, created at, description name, follower count, member count
Hashtag	A hashtag is a metadata tag that is prefaced by "#". It is used to link tweets with the same theme together.	hashtag name

Table 5: Relations in the TwiBot-22 heterogeneous graph.

Relation	Source Entity	Target Entity	Description
following	user	user	user A follows user B
follower	user	user	user A is followed by user B
post	user	tweet	user A posts tweet B
pin	user	tweet	user A pins tweet B
like	user	tweet	user A likes tweet B
mention	tweet	user	tweet A mentions user B
retweet	tweet	tweet	tweet A retweets tweet B
quote	tweet	tweet	tweet A quotes tweet B with comments
reply	tweet	tweet	tweet A replies to tweet B
own	user	list	user A is the creator of list B
member	user	list	user A is a member of list B
follow	user	list	user A follows list B
contain	list	tweet	list A contains tweet B
discuss	tweet	hashtag	tweet A discussed hashtag B

A TwiBot-22 Details

A.1 Entities and Relations

TwiBot-22 collects four types of entities on the Twitter social network: user, tweet, list, and hashtag. The detailed information of these entities is shown in Table 4 while a complete list of all relation types in TwiBot-22 is presented in Table 5.

A.2 Data Collection Details

The complete process of data collection. For the first stage of user network collection, we adopt *@NeurIPSConf* as the starting user. We use the Twitter API to retrieve 1,000 followers and 1,000 followees as the user's neighborhood for BFS expansion. We randomly adopt one of the two sampling strategies (distribution diversity or value diversity) and randomly select one metadata from Table 6 to include 6 users from its neighborhood into the TwiBot-22 dataset. We then randomly select one unexpanded user in TwiBot-22 for a new round of neighborhood expansion. For the second stage of heterogeneous graph building, we first collect 1,000 tweets for the user in the user network, and

Table 6: User metadata adopted in diversity-aware sampling.

Metadata Name	Description	Type
active days	days between user creation time and collected time	numerical
following count	number of user followings	numerical
followers count	number of user followers	numerical
tweet count	number of user tweets	numerical
listed count	number of user lists	numerical
verified	whether the user is verified or not	true-or-false
homepage url	whether user has urls in homepage or not	true-or-false

Table 7: Statistics of TwiBot-22.

Item	Value	Item	Value	Item	Value
entity type	4	post	88,217,457	following	2,626,979
relation type	14	pin	347,131	follower	1,116,655
user	1,000,000	like	595,794	contain	1,998,788
hashtag	5,146,289	mention	4,759,388	discuss	66,000,633
list	21,870	retweet	1,580,643	bot	139,943
tweet	88,217,457	quote	289,476	human	860,057
user metadata	17	reply	1,114,980	entity	92,932,326
hashtag metadata	2	own	21,870	relation	170,185,937
list metadata	8	member	1,022,587	max degree	270,344
tweet metadata	20	follow	493,556	verified user	95,398

200 tweets for the user expanded from the user network. We collect the pinned tweet and the recent 100 liked tweet of each user in TwiBot-22. For each tweet we collect now, we collect the tweets it retweets, quotes, or replies and the users it mentions. We collect a user's recent 100 lists with the newest 100 members, followers, and tweets. We collect all hashtags in the tweets and search for more tweets related to a hashtag using Twitter API. Finally, we make sure that the creator of each tweet is collected and collect 40 tweets of these users according to Ng et al. [2022] for stable benchmarking

Data collection time. The first stage of user network collection is conducted from January 20th, 2022 to February 1st, 2022. The second stage of heterogeneous graph building is conducted from February 1st, 2022 to March 15th, 2022.

A.3 Expert Annotation Details

We invite 17 researchers in our group who are active Twitter users, are familiar with bot detection literature, and have conducted experiments with the TwiBot-20 datasets. We then assign each Twitter user in TwiBot-22 to 5 different experts and ask them to evaluate whether the user is a human, a bot, or not sure. We use majority voting to obtain the expert annotations of these 1,000 users, which are then leveraged to guide the weak supervision leaning process.

A.4 Dataset Statistics

Table 7 presents important statistics about the TwiBot-22 benchmark.

A.5 Annotation Quality Study Details

To compare the annotation quality of TwiBot-22 and TwiBot-20, we ask 6 researchers to participate in an expert study. They are familiar with Twitter bot detection research and most of them have previously published on this topic. Specifically, we randomly select 500 users from TwiBot-20 and TwiBot-22 respectively and assign each user to 3 experts. We then ask them to evaluate each user as "definitely bot", "likely bot", "not sure", "likely human", and "definitely human". Based on their evaluations, we calculate the accuracy and F1-score between expert opinions and dataset labels. We also report the Randolph's Kappa Coefficient [Randolph, 2005], which models agreement between experts in Figure 2(c).

A.6 Other TwiBot-22 Details

Dataset documentation. We encourage the readers to refer to the TwiBot-22 evaluation framework (`https://github.com/LuoUndergradXJTU/TwiBot-22`) for the documentation of TwiBot-22 and the TwiBot-22 evaluation framework.

Intended use. TwiBot-22 should be used for research in Twitter bot detection and social network analysis.

Relevent URLs. We list all TwiBot-22 URLs in the following.

- **Official TwiBot-22 website** (`https://twibot22.github.io/`) is the main reference of TwiBot-22, presenting our dataset, paper, evaluation framework, and contact information.
- **TwiBot-22 repository** (`https://github.com/LuoUndergradXJTU/TwiBot-22`) hosts implemented codes for dataset preprocessing and 35 Twitter bot detection methods. The repository is well documented to facilitate reproducible research.
- **TwiBot-22 dataset** will be permanently hosted on our group Google Drive account (`https://drive.google.com/drive/folders/1YwiOUwtl8pCd2GD97Q_WEzwEUtSPoxFs?usp=sharing`).

Hosting and maintenance. The TwiBot-22 dataset will be hosted via Google Drive and regularly maintained by our research group. The TwiBot-22 evaluation framework will be hosted on GitHub. Our team will continue to add new datasets and baselines with the help of the research community.

Licensing. The TwiBot-22 dataset uses the CC BY-NC-ND 4.0 license. Implemented code in the TwiBot-22 evaluation framework uses the MIT license.

Author statement. We bear all responsibility in case of violation of rights, etc., and confirmation of the data license.

Limitations. One minor limitation of TwiBot-22 is that we do not download and store user media (images and videos) in TwiBot-22, while these multimedia content might be useful for bot detection. However, if researchers do deem multimedia content as necessary for bot detection, they can download with media links in TwiBot-22 by themselves.

Potential negative societal impact. Although TwiBot-22 and the TwiBot-22 evaluation framework are designed to facilitate bot detection research and improve bot detection models, it might be abused by bot operators to examine the characteristics of bots that evade detection, and thus designing bot algorithms that are more evasive. We need to make sure that the TwiBot-22 dataset and evaluation framework should not be abused to design advanced Twitter bots.

B Experiment Details

B.1 Baseline Details

We briefly describe each of the 35 Twitter bot detection baseline methods:

- **SGBot** [Yang et al., 2020]. SGBot is proposed to address the scalability and generalization problem in Twitter bot detection. SGBot leverages 8 types of user metadata such as status count and 12 derived features such as tweet frequency and adopts random forest to identify bot.
- **Kudugunta *et al.*** [Kudugunta and Ferrara, 2018]. The baseline addresses two challenges, account-level classification and tweet-level classification. In the task of account-level classification, the baseline introduces a technique that combines synthetic minority oversampling (SMOTE) with undersampling techniques. In the task of tweet-level classification, the baseline introduces an architecture called contextual LSTM.

- **Hayawi *et al.*** [Hayawi et al., 2022]. DeeProBot, which is short for Deep Profile-based Bot detection Framework, utilizes different types of features including user's numerical or binary metadata and user's description, making the model more comprehensive. DeeProBot uses GLoVe word embeddings to get the embedding of the textual information. LSTM and dense layers are then used to learn user representations for bot detection.
- **BotHunter** [Beskow and Carley, 2018]. In this work, the features are composed of user attributes, network attributes, contents, and timing information. After extracting the above features, random forest is exploited as the classifier.
- **NameBot** [Beskow and Carley, 2019]. NameBot utilizes twitter username as the only classification basis and extracts numerical features such as TF-IDF. The extracted numeric features are then exploited as input of a linear regression classifier. While achieving great good accuracy on training set, this basline shows poor transferbility on new datasets.
- **Abreu *et al.*** [Abreu et al., 2020]. This model chooses five essential Twitter user features to conduct relative experiments. They calculated accuracy, AUC, recall and F1-score on several existing datasets with four machine learning algorithms.
- **Cresci *et al.*** [Cresci et al., 2016]. This method encodes different action types with different characters, thus representing usernames by strings. Users who share the longest common substring are considers as bots have similar behaviors.
- **Wei *et al.*** [Wei and Nguyen, 2019]. This paper use pre-trained GloVe word vectors on Twitter as word embedding. Multiple layers of bidirectional LSTMs are used for bot detection.
- **BGSRD** [Guo et al., 2021a]. This model utilizes BERT and GCN to achieve social bots detection. In specific, this paper use word and user description as graph nodes, there are no connection within users and words. In training steps, BGSRD first uses BERT (Roberta) to process users' description, the output of BERT layer is the initial feature of graph nodes, and the initial feature of words node is zeros. The final prediction is the combination of BERT and GAT output.
- **RoBERTa** [Liu et al., 2019]. This baseline leverages pre-trained language model RoBERTa to encode user tweets and descriptions, then feed them to an MLP to distinguish bots from human.
- **T5** [Raffel et al., 2020]. In this baseline, we use pretained language model T5-small to encode tweets and descriptions, then feed them into an MLP classifier.
- **Efthimion *et al.*** [Efthimion et al., 2018]. This paper leverages a wide range of users' features including length of user names, reposting rate, temporal patterns, sentiment expression, followers-to-friends ratio, and message variability for bot detection. Logistic regression and support vector machine are applied successively for profile and account activity analysis. Levenshtein distance is applied for text mining.
- **Kantepe *et al.*** [Kantepe and Ganiz, 2017]. This paper explores the importance of features by comparing sixty-two different features related to Twitter account properties and tweet contents, and selects the most important eleven features through experimental comparison. The classification task was then performed using 62 or 11 features as input with classifier like GBDT or SVM.
- **Miller *et al.*** [Miller et al., 2014]. Miller et al. proposed a clustering based method to detect spam accounts in twitter. Specifically, they exploit classic clustering algorithms such as DBSCAN and K-MEANS on human accounts to obtain human clusters. In test phase, users whose clusters can not be filed under any of the existing human clusters are consider as bots.
- **Varol *et al.*** [Varol et al., 2017]. User metadata, tweet content, friends, sentiment and network statistics are adapted as user's features, then the extracted features are utilized as inputs of a random forest model.
- **Kouvela *et al.*** [Kouvela et al., 2020]. This baseline leverages user features and content features from each user and classifies users with random forest. Specifically, it uses 36 features from each account and the content features are extracted from the latest 20 tweets.
- **Santos *et al.*** [Ferreira Dos Santos et al., 2019]. This baseline extracts 16 features from users' tweets and descriptions and feed the features into a decision tree for classification.
- **Lee *et al.*** [Lee et al., 2011]. This method introduces the social honeypots to attract bot users by manipulating the honeypot users' tweets frequency and social network structure. After analyzing the data collected by social honeypots, 18 features are selected and fed into the random forest classifier.

- **LOBO** [Echeverrï£¡ a et al., 2018]. This baseline extracts 19 features (26 on Twibot-22) from each user and adopts random forest for classification.
- **Moghaddam *et al.*** [Moghaddam and Abbaspour, 2022]. This model combines profile-based features and friendship preference features, which compares the distribution of followers' features and sub-population of accounts.
- **Alhosseini *et al.*** [Ali Alhosseini et al., 2019]. This model uses age, statuses_count, account length name, followers_count, friends_count and favourites_count as user features and feed them into a GCN layer to identify bot users.
- **Knauth *et al.*** [Knauth, 2019]. This paper extracts features from user's meta data, tweets, user behavior, and feeds these features into Adaboost classifier.
- **FriendBot** [Beskow and Carley, 2020]. This paper introduces network metrics into twitter bot detection tasks. Specifically, they construct a 2-hop ego network for each twitter account based on four types of relations: following, retweet, mention, and reply. They collect account features based on metrics of these ego networks. Finally, they exploit random forest algorithm for classification.
- **SATAR** [Feng et al., 2021a]. SATAR is a self-supervised representation learning framework of Twitter users. SATAR jointly uses semantic, property, and neighborhood information and adopts a co-influence module to aggregate these information. SATAR considers the follower count as self-supervised label to pre-train parameters and fine-tune parameters in bot detection task.
- **Botometer** [Yang et al., 2022]. Botometer (formerly BotOrNot) is a public website to check the activity of a Twitter account and give it a score, where higher scores mean more bot-like activity. Botometer's classification system leverages more than 1,000 features using available meta-data and information extracted from interaction patterns and content.
- **Rodriguez-Ruiz *et al.*** [Rodríguez-Ruiz et al., 2020]. This paper designs a one-class classification model, which uses the social network and tweet information of Twitter users to extract 13 features for feature engineering, and the model has also achieved good classification results.
- **GraphHist** [Magelinski et al., 2020]. The authors design a new graph classifier based on histogram and customized backward operator. By exploiting the ego-graph of twitter users, bot detection can be solved by utilizing the proposed graph-level classifier.
- **EvolveBot** [Yang et al., 2013]. This method designs 11 robust features, together with 7 efficient features to combat evasion tatics of spammers.
- **Dehghan *et al.*** [Dehghan et al., 2022]. This baseline combines the profile features, text features, and graph features for bot detection. After obtaining account representations through Deepwalk and struc2vec, XGBclassifier is applied to identify bot users.
- **GCN** [Kipf and Welling, 2016]. GCN aggregates features from neighbors equally and learns representation for each user. These representations are passed to an MLP for classification. The initial user features are identical with BotRGCN.
- **GAT** [Veličković et al., 2018]. Graph Attention Network (GAT) introduces attention mechanism to GNN models, making it capable of distinguishing the importance of neighboring users in aggregation. Same as GCN, it can learn user representations and feed them into an MLP for classification. The initial user features are identical with BotRGCN.
- **HGT** [Hu et al., 2020]. Heterogeneous Graph Transformers (HGT) is a dedicated heterogeneous GNN that mainly consists of two modules, heterogeneous mutual attention and heterogeneous message passing. Heterogeneous mutual attention considers the edge type and source and target node type when calculating attention scores. Heterogeneous message passing module incorporates the source node type and the edge dependency in passed messages. The initial user features are identical with BotRGCN.
- **simpleHGN** [Lv et al., 2021]. SimpleHGN is a simple yet effective GNN for heterogeneous graph inspired by the GAT. SimpleHGN adopts three strategies to enhance GAT, learnable embedding for each edge-type, node-level and edge-level residual connections as well as the L2 regularization on output representations. The initial user features are identical with BotRGCN.
- **BotRGCN** [Feng et al., 2021b]. BotRGCN utilizes the text information from user descriptions and tweets, as well as numerical and categorical user property information. Then BotRGCN constructs a heterogeneous graph from the Twitter network based on user relationships and relational graph convolutional networks (R-GCN) is applied to learn user representations for bot detection tasks.

Table 8: Average model performance (F1-score) and standard deviation of 35 baseline methods on 9 datasets. **Bold** and underline indicate the highest and second highest performance. The F, T, and G in the "Type" column stands for feature, text, and graph. Cresci et al. and Botometer are deterministic methods without standard deviation. / indicates that the dataset could not support the baseline. - indicates that the baseline could not scale to the largest TwiBot-22 dataset.

Method	**Type**	**C-15**	**G-17**	**C-17**	**M-18**	**C-S-18**	**C-R-19**	**B-F-19**	**TwiBot-20**	**TwiBot-22**
SGBot	F	77.9 (±0.1)	72.1 (±1.2)	94.6 (±0.2)	99.5 (±0.0)	82.3 (±0.1)	82.7 (±1.7)	49.6 (±3.4)	84.9 (±0.4)	36.6 (±0.2)
Kudugunta *et al.*	F	75.3 (±0.2)	49.8 (±2.1)	91.7 (±0.2)	94.5 (±0.3)	50.9 (±0.4)	49.2 (±1.3)	49.6 (±8.2)	47.3 (±1.4)	51.7 (±0.0)
Hayawi *et al.*	F	85.6 (±0.0)	34.7 (±0.1)	93.8 (±0.0)	91.5 (±0.0)	60.8 (±0.1)	60.9 (±0.0)	20.5 (±0.1)	77.1 (±0.0)	24.7 (±0.1)
BotHunter	F	97.2 (±1.0)	69.2 (±1.0)	91.6 (±0.1)	**99.6** (±0.0)	82.2 (±0.2)	82.9 (±1.9)	49.6 (±3.1)	79.1 (±0.4)	23.5 (±0.1)
NameBot	F	83.4 (±0.0)	44.8 (±0.0)	85.7 (±0.0)	91.6 (±0.0)	61.1 (±0.0)	67.5 (±0.0)	38.5 (±0.0)	65.1 (±0.1)	0.5 (±0.0)
Abreu *et al.*	F	76.4 (±0.1)	66.7 (±0.1)	95.0 (±0.1)	97.9 (±0.1)	76.9 (±0.1)	**83.5** (±0.1)	**53.8** (±0.1)	77.1 (±0.1)	53.4 (±0.1)
Cresci *et al.*	T	1.17	/	22.8	/	/	/	/	13.7	-
Wei *et al.*	T	82.7 (±2.2)	/	78.4 (±1.7)	/	/	/	/	57.3 (±3.1)	53.6 (±1.3)
BGSRD	T	90.8 (±0.6)	35.7 (±32.6)	86.3 (±0.0)	90.5 (±1.0)	58.2 (±12.0)	41.1 (±13.0)	13.0 (±13.0)	70.0 (±2.6)	21.1 (±29.0)
RoBERTa	T	95.8 (±0.1)	/	94.3 (±0.1)	/	/	/	/	73.1 (±0.5)	20.5 (±1.7)
T5	T	89.3 (±0.2)	/	92.3 (±0.1)	/	/	/	/	70.5 (±0.3)	20.2 (±2.0)
Efthimion *et al.*	FT	94.1 (±0.0)	5.2 (±0.0)	91.8 (±0.0)	95.9 (±0.0)	68.2 (±0.0)	71.7 (±0.0)	0.0 (±0.0)	67.2 (±0.0)	27.5 (±0.0)
Kantepe *et al.*	FT	78.2 (±1.4)	/	79.4 (±1.3)	/	/	/	/	62.2 (±2.1)	**58.7** (±1.6)
Miller *et al.*	FT	83.8 (±0.0)	59.9 (±0.0)	86.8 (±0.1)	91.1 (±0.0)	56.8 (±0.0)	43.6 (±0.0)	0.0 (±0.0)	74.8 (±0.3)	45.3 (±0.0)
Varol *et al.*	FT	94.7 (±0.4)	/	/	/	/	/	/	81.1 (±0.5)	27.5 (±0.3)
Kouvela *et al.*	FT	98.2 (±0.4)	66.6 (±1.7)	99.1 (±0.1)	98.2 (±0.1)	80.4 (±0.2)	81.1 (±1.0)	28.1 (±5.3)	86.5 (±0.3)	30.0 (±0.0)
Santos *et al.*	FT	78.8 (±0.0)	14.5 (±0.0)	83.0 (±0.0)	92.4 (±0.0)	65.2 (±0.0)	75.7 (±0.0)	21.0 (±0.0)	60.3 (±0.0)	-
Lee *et al.*	FT	98.6 (±0.1)	67.8 (±1.8)	**99.3** (±0.0)	97.9 (±0.1)	82.5 (±0.4)	82.7 (±1.8)	50.3 (±3.2)	80.0 (±0.5)	30.4 (±0.2)
LOBO	FT	**98.8** (±0.3)	/	97.7 (±0.2)	/	/	/	/	80.8 (±0.2)	38.6 (±0.2)
Moghaddam *et al.*	FG	73.9 (±0.2)	/	/	/	/	/	/	79.9 (±0.7)	32.1 (±0.0)
Alhosseini *et al.*	FG	92.2 (±0.4)	/	/	/	/	/	/	72.0 (±0.5)	38.1 (±5.9)
Knauth *et al.*	FTG	91.2 (±0.0)	39.1 (±0.0)	93.4 (±0.0)	91.3 (±0.0)	**94.0** (±0.0)	54.2 (±0.0)	41.3 (±0.0)	85.2 (±0.0)	37.1 (±0.0)
FriendBot	FTG	97.6 (±0.8)	/	87.4 (±0.5)	/	/	/	/	80.0 (±0.3)	-
SATAR	FTG	95.0 (±0.3)	/	/	/	/	/	/	86.1 (±0.7)	-
Botometer	FTG	66.9	**77.4**	96.1	46.0	79.6	79.0	30.8	53.1	42.8
Rodrifuez-Ruiz *et al.*	FTG	87.7 (±0.0)	/	85.7 (±0.0)	/	/	/	/	63.1 (±0.1)	56.6 (±0.0)
GraphHist	FTG	84.5 (±8.2)	/	/	/	/	/	/	67.6 (±0.3)	-
EvolveBot	FTG	90.1 (±2.0)	/	/	/	/	/	/	69.7 (±0.5)	14.1 (±0.1)
Dehghan *et al.*	FTG	88.3 (±0.0)	/	/	/	/	/	/	76.2 (±0.0)	-
GCN	FTG	97.2 (±0.0)	/	/	/	/	/	/	80.8 (±0.0)	54.9 (±0.0)
GAT	FTG	97.6 (±0.0)	/	/	/	/	/	/	85.2 (±0.0)	55.8 (±0.0)
HGT	FTG	96.9 (±0.2)	/	/	/	/	/	/	**88.2** (±0.2)	39.6 (±2.1)
SimpleHGN	FTG	97.5 (±0.4)	/	/	/	/	/	/	**88.2** (±0.2)	45.4 (±0.4)
BotRGCN	FTG	97.3 (±0.5)	/	/	/	/	/	/	87.3 (±0.7)	57.5 (±1.4)
RGT	FTG	97.8 (±0.2)	/	/	/	/	/	/	88.0 (±0.4)	42.9 (±0.5)

- **RGT** [Feng et al., 2022]. Relational Graph Transformers is a GNN framework that uses graph transformers and semantic attention network to model the intrinsic influence heterogeneity and relation heterogeneity in Twittersphere. Specifically, RGT first learns users' representation under each relation with graph transformers, then it aggregate representations from all relations using the semantic attention network.

B.2 F, T, or G?

We categorize baseline methods into F, T, or G with the following rules:

- If the baseline leverages user metadata and conduct feature engineering, the baseline is F.
- If the baseline leverages the content of tweets and user description texts, the baseline is T.
- If the baseline leverages the network structure of Twitter, the baseline is G.

Baseline methods may check multiple boxes and have multiple types. For exmample, BotRGCN [Feng et al., 2021b] is F since it selects user metadata and encode users as feature vectors. BotRGCN is T since it encodes user tweets and description with pre-trained RoBERTa. BotRGCN is G since it constructs a relational graph and adopts relational graph neural networks for bot detection. As a result, BotRGCN has the type of FTG.

Table 9: Example hashtags in the five hashtag-based sub-communities.

ID	Example Hashtags
1	#Christ, #Taliban, #Kabul, #Germany, #EU, #manufacturer, #Manchester, #Covid, #covid, #bitcoin, #Ukraine, #Kyiv, #Iowa, #farm, #health, #bullying, #Putin, #gerrymandering, #Covid19, #Labour
2	#cybersecurity, #CVE, #GCP, #marketing, #datacenter, #OSINT, #SMEs, #aerospace, #innovation, #science, #exoplanet, #log4j, #conservation, #farming, #biology, #chemistry, #agriculture, #growth, #aging, #dementia
3	#Curitiba, #Colombia, #inversiones, #emprender, #emprendedor, #negocios, #liderazgo, #bici, #ciclismo, #RRSS, #correr, #sueños, #metas, #familia, #inversión, #Fortalecimiento, #emprendimiento, #Familia, #éxito, #ventas
4	#Vimeo, #Industry, #Anonymous, #iHeartRadio, #Biomass, #Contest, #Books, #Humor, #Memoir, #Storytelling, #Butterfly, #Art, #Canvas, #Handbag, #Tshirt, #Kidney, #Passion, #Quarantine, #Whitelist, #PMC
5	#UCL, #coach, #FF, #USMNT, #Coventry, #Orpheus, #CRO, #Sydney, #Houston, #Jordan, #Buffalo, #UBC, #writer, #Shona, #Christchurch, #Antigua, #Sonny, #Gladstone, #500th, #Philips

Table 10: Statistics of the 10 sub-communities.

Sub-communities	0	1	2	3	4	5	6	7	8	9
# Human	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
# Bot	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
# User	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000
# Tweet	969,979	942,020	1,099,962	989,536	1,083,655	1,156,640	1,333,018	1,138,480	1,151,362	1,142,717
# Edge	1,116,208	1,120,637	1,245,190	1,167,285	1,249,535	1,535,397	1,924,616	1,508,054	1,511,824	1,526,627

B.3 F1-score Results

We re-implement 35 Twitter bot detection baselines and evaluate them on 9 representative datasets and benchmarks. We present their detection accuracy in Table 2 and F1-score in Table 8.

B.4 Graph Component Removal Details

We remove the graph component in graph-based methods to examine the role of graphs in Twitter bot detection and present results in Table 3. We provide details about how graphs are removed from each baseline as follows:

- **Alhosseini *et al.***. We remove the GCNs while using two MLP layers with user features.
- **Moghaddam *et al.***. We remove 11 graph-based features from the "friend preference" section.
- **Knauth *et al.***. We remove 2 graph-based features, namely friend count and follower count.
- **EvolveBot**. We remove 4 graph-based feature extracted with the help of neighbor information.
- **BotRGCN** and **RGT**. We remove the R-GCN and RGT while using two MLP layers with user features for bot detection. BotRGCN and RGT use the same user features so that their "w/o graph" results are identical.

B.5 Generalization Study Details

To evaluate existing methods and their ability to generalize on unseen data, we identify 10 sub-communities in the TwiBot-22 network and conduct experiments in Figure 3. Specifically, we firstly select 5 closely connected sub-communities around @*BarackObama*, @*elonmusk*, @*CNN*, @*NeurIPSConf*, and @*ladygaga*. These five users feature different interest domains and their neighborhood represents different aspects of the Twitter network. In addition, we use K-means to cluster the word2vec [Mikolov et al., 2013] representations of hashtags and identify users tweeting about similar hashtags into 5 sub-communities. Examples of these hashtags in these sub-communities are presented in Table 9. The statistics of the 10 sub-communities are presented in Table 10.

B.6 Computation Details

We ran all experiments on a server with 8 GeForce RTX 2080 Ti GPUs. We run each experiment for five times and report the average model performance as well as standard deviation.

Table 11: We remove labeling functions in the annotation process and compare their results with the full annotation model.

labeling function	bot->bot	bot->human	human->human	human->bot	changed percentage
w/o adaboost	77,050	49,965	869,317	3,986	5.36%
w/o random forest	117,191	9,524	841,531	31754	4.13%
w/o MLP	109,152	17,563	796,159	77,126	9.46%
w/o GCN	120,925	5,790	833,776	39,509	4.54%
w/o GAT	123,397	3,318	824,436	48,849	5.22%
w/o RGCN	123,676	3,042	819,293	53,970	5.70%
w/o verify	122,177	4,538	873,101	184	0.47%
w/o keywords	123,880	2,835	840,142	33,143	3.60%

Table 12: We use the training set and validation set in TwiBot-22 while using the expert labels as the test set. We used 6 baseline methods for a quick evaluation. Test 1 indicates using only the 1,000 manually annotated users in Section 3.2, and test 2 indicates using only the 500 manually annotated users in Section 3.3.

Model	test set 1				test set 2			
	Accuracy	F1-score	Precision	Recall	Accuracy	F1-score	Precision	Recall
Moghaddam *et al.*	89.41 (±0.30)	24.98 (±2.72)	16.57 (±1.97)	50.79 (±4.25)	83.93 (±0.28)	18.49 (±0.95)	11.58 (±0.59)	45.94 (±3.35)
SGBot	91.87 (±0.11)	47.43 (±1.21)	76.16 (±2.31)	34.48 (±1.56)	87.42 (±0.31)	26.00 (±2.80)	54.55 (±2.80)	17.11 (±2.28)
BotHunter	91.44 (±0.12)	40.39 (±0.32)	78.28 (±3.11)	27.24 (±0.52)	85.63 (±0.31)	23.38 (±1.55)	73.67 (±9.81)	13.95 (±1.18)
GAT	91.14 (±0.45)	47.00 (±2.92)	64.83 (±4.31)	36.95 (±3.04)	84.93 (±0.23)	30.47 (±2.64)	55.64 (±2.02)	21.05 (±2.46)
BotRGCN	88.74 (±0.29)	65.89 (±1.62)	79.82 (±2.53)	56.23 (±3.24)	85.59 (±0.68)	55.45 (±2.77)	67.45 (±2.74)	47.17 (±3.65)
RGT	92.80 (±0.45)	23.39 (±4.61)	58.33 (±11.78)	16.44 (±2.98)	87.10 (±1.19)	38.02 (±7.21)	58.50 (±10.18)	28.57 (±6.68)

B.7 Annotation Bias Test

To study the effect of individual labeling function, we remove each of them and examine how many labels have changed in the snorkel-based annotation process, as is shown in Table 11.

Experiment results on using the expert labels as the test set is presented in Table 12.

B.8 Implementation Details

The TwiBot-22 evaluation framework is built with help of many valuable scientific artifacts, including pytorch [Paszke et al., 2019], pytorch lightning [Falcon and The PyTorch Lightning team, 2019], pygod [Liu et al., 2022], transformers [Wolf et al., 2020], pytorch geometric [Fey and Lenssen, 2019], sklearn [Pedregosa et al., 2011], gensim [Řehůřek and Sojka, 2010], spacy [Honnibal et al., 2020], tweepy [Roesslein, 2009], pandas [McKinney et al., 2011], numpy [Harris et al., 2020], vaderSentiment [Hutto and Gilbert, 2014], and imbalanced-learn [Lemaître et al., 2017].