Thesis Dissertation

A COMPARATIVE ANALYSIS OF FEATURES AND METHODS FOR GITHUB BOT DETECTION

Michalis Chiripi

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2026

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

A Comparative Analysis of Features and Methods for GitHub Bot Detection

Michalis Chiripi

Eleni Constantinou

Daniel Garcia Rodriguez

Thesis submitted in partial fulfillment of the requirements for the award of degree of bachelor's in computer science at University of Cyprus.

May 2026

I declare that this dissertation and any accompanying software are my own original work, conducted in full compliance with the University of Cyprus Code of Conduct for Research (https://www.ucy.ac.cy/legislation/kodikas-deontologias-kai-politikes/), and with full accountability on my part for the accuracy, integrity, and validity of all content."

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my family for their unwavering support and encouragement throughout my academic journey. Their constant belief in me, along with the sacrifices they made to provide me with this opportunity, has been invaluable. I am truly thankful for everything they have done.

Secondly I would also like to sincerely thank my friends and classmates for their support over the past years. Their encouragement, understanding, and shared experiences helped me overcome challenges and stay motivated, even during the most demanding times. I am grateful to have gone through this journey alongside such an amazing group of people.

More importantly a special thanks goes to my supervisors, Eleni Constantinou and Daniel Garcia Rodriguez, for their guidance, valuable insights, and continuous support throughout the development of this thesis.

ABSTRACT

Software bots have become a common part of software development on platforms such as GitHub. These accounts automate tasks such as dependency updates, issue management, continuous integration checks, release preparation, and repository maintenance. Although bots are useful for development teams, they can also create problems for software engineering research when automated activity is analyzed as if it was human activity.

This thesis studies GitHub bot detection by comparing two account-level detection approaches: BotHunter and RABBIT. BotHunter uses profile, activity, and text-similarity features, while RABBIT focuses on activity sequences and temporal behavior. The aim of this work is to understand which features are the strongest indicators for distinguishing bot accounts from human accounts, and how the amount of available data affects BotHunter's predictions.

The analysis is based on a ground-truth list of GitHub contributors provided by the RABBIT/BIMBAS authors. The tools were adapted to export their extracted features, and the features were analyzed using the Mann-Whitney U test, Cliff's delta, and model-based feature importance. A prefix experiment was also performed for BotHunter using 25%, 50%, 75%, and 100% of each account's lifetime data.

The results show that BotHunter is mainly driven by profile-oriented features, especially account login, followers, following, and account name. RABBIT, in contrast, relies more on behavioral and temporal activity features. The prefix experiment shows that BotHunter performs well early when bots have clear naming signals, but additional lifetime data does not substantially improve its predictions for unnamed bots. The study concludes with implications for researchers, repository maintainers, and future bot detection tools.

Contents

Section 1

Introduction

1.1 Research and Motivation

GitHub has become one of the most important platforms for collaborative software development. It hosts millions of repositories and supports a broad range of development activities, including source code management, issue tracking, pull request review, release preparation, and project maintenance. As open-source and industrial software projects increasingly rely on automation, bots have become a visible and often essential part of this ecosystem. Dependency bots, continuous integration bots, moderation bots, release bots, and notification bots help development teams reduce repetitive work, react faster to changes, and maintain repositories more efficiently[12].

At the same time, the growing presence of bots creates a methodological problem. Many studies in mining software repositories assume that the observed activity on GitHub reflect human collaboration. When a substantial proportion of that activity is or might have been generated by bots, the resulting datasets no longer describe only human behavior. This can distort measurements of productivity, communication, workload, review practices, and contribution patterns.

Bot detection on GitHub is therefore not only a classification task but also a step for more reliable software engineering analysis. However, the problem is more difficult than it may appear. Some automated accounts are clearly labeled as bots or GitHub Apps, but many others are ordinary user accounts that behave in a bot like way without being explicitly marked. These accounts can commit code, post comments, open pull requests, or interact with several repositories in a way that resembles human activity at first glance. As a result, simple tells such as checking whether an account contains the word 'bot' in its username are not sufficient on their own.

Prior work has proposed several approaches to address this problem, but these approaches differ substantially in their assumptions and in the kinds of features they rely on. Some models focus on commits, some on comments, and others aim to aggregate multiple activity types. Some require long historical records, whereas others are designed to work with a smaller event window.

1.2 Research Questions

The first objective of this thesis is to analyze feature importance for two representative GitHub bot detection approaches: BotHunter and RABBIT. These tools were selected because they have different designs. BotHunter combines account profile information and textual similarity whereas RABBIT, focuses more on activity sequence and temporal information derived from recent GitHub events. This contrast makes them particularly suitable for a comparative study on feature ranking.

The second objective is to examine how the prediction and confidence is changed when different amounts of data are given to the model. The full vision of the project includes experiments with lifetime histories and partial history prefixes and seeing how much data does a model need in order to make a more accurate prediction.

Based on these objectives, the thesis is guided by the following research questions:

• RQ1: Which features are the strongest indicators for distinguishing GitHub bot accounts from GitHub human accounts?

• RQ2: How much data does a model need in order to provide a better prediction, and how does the prediction change when less or more data is available?

1.3 Thesis Structure

The remainder of this thesis is organized as follows. Section 2 presents the background and related work on GitHub bot detection approaches, focusing on BoDeGHa, BIMAN, BotHunter, and RABBIT. Section 3 describes the methodology, including dataset preparation, JSON collection, tool adaptation, and statistical analysis. Section 4 presents the feature-ranking results for BotHunter and RABBIT. Section 5 reports the prefix experiment used to evaluate how BotHunter predictions change with different amounts of lifetime data. Section 6 discusses the contributions, implications and limitations of the results. Finally section 7 concludes the thesis by answering the research questions and outlining directions for future work.

Section 2

Background & Related Work

2.1 Bot Detection Approaches

GitHub distinguishes between different actor types, such as Users, Bots and Organizations. However, this distinction is not sufficient for bot detection because many automated accounts are still registered as normal User accounts. That implies that a GitHub account can appear as a human but still being used to automate repetitive tasks. This makes bot detection necessary for studies since it can produce wrong data. Therefore this section focuses on four bot identification approaches: BoDeGHa, BIMAN, BotHunter, and RABBIT [1, 6, 8, 10]. It then situates these tools within broader work on software bots and bot detection.

2.1.1 BoDeGHa

BoDeGHa (Bot Detection on GitHub) is one of the earlier approaches for identifying bot accounts on GitHub. It primarily focuses on analyzing textual patterns in user-generated content, particularly issue and pull request comments [10].
The key idea behind BoDeGHa is that bots tend to generate repetitive or highly similar text. For example, a bot that automatically responds to issues may repeatedly post messages with identical or slightly modified content. By computing similarity metrics between comments,

BoDeGHa can identify accounts that exhibit high levels of textual repetition. Accounts such as moderation bots, notification bots, and dependency management bots often respond through standard message templates, and comment similarity can capture this regularity well While this approach is effective for detecting certain types of bots, particularly those that rely heavily on automated messaging, it has several limitations. First, it focuses primarily on comment based activity and does not consider other types of interactions such as commits or repository activity. Second, it may fail to detect bots that generate diverse or context dependent messages. As a result, its applicability is limited to specific categories of bots, thus resulting in its exclusion from the final experimental analysis.

2.1.2 BIMAN

BIMAN was introduced to detect bots that commit code. In contrast to comment oriented approaches, BIMAN studies commit related evidence such as author names, commit messages, file associations, and project information [8]. This makes it especially relevant for bots whose main observable behavior is producing commits rather than participating in issue discussions or pull request conversations.

The contribution of BIMAN is important for two reasons. First, it showed that commit-oriented automation can be detected at scale and that bots leaving commit histories often exhibit patterns that differ from those of human developers. Second, it broadened the understanding of the GitHub bot detection problem by demonstrating that different activity channels require different signals. A bot that primarily commits code may be almost invisible to a comment based detector, just as a comment heavy moderation bot may not be well described by commit only evidence.

BIMAN was examined in the background study but was excluded from the final experimental analysis. The main reason for this exclusion is that BIMAN relies heavily on commit level data and, in practice, its proper reproduction requires access to data derived from World of Code. This created a significant limitation for this thesis, since such data was not readily available in the experimental environment used for this work

2.1.3 BotHunter

BotHunter is designed to detect software bots on GitHub mostly at account level, with the addition of activity and text similarity[1]. Unlike earlier approaches that focus on a single activity type, such as commits or issue comments, BotHunter aims to provide a generalized solution by combining multiple sources of information, including user profile data, activity

patterns, and textual behavior. This makes it particularly suitable for identifying both commit based bots and issue/pull request bots.

BotHunter takes each GitHub account and classifies it as either a human or a bot. To achieve this, the approach extracts a set of 18 carefully selected features that capture different aspects of user behavior. These features are grouped into three main dimensions: profile information, account activity , and text similarity. The use of these feature categories allows BotHunter to build a representation of each account, improving its ability to distinguish between automated and human behavior.

The first category, profile information features, focuses on static attributes of GitHub accounts. These features include the account login (username), account name, account bio, number of followers, number of followings, and whether the account is tagged as a bot. These features are particularly useful because many bots exhibit identifiable patterns in their profiles. For example, bot accounts often include the substring “bot” in their login or name, explicitly describe themselves as bots in their bio, or have very limited social interaction. Additionally, bot accounts tend to have fewer followers compared to humans, although popular bots may still accumulate a large following. These profile based features serve as strong initial indicators and, according to the study, are among the most important predictors of account type.

The second category, account activity features, captures how users interact with GitHub over time. These features show both the volume and diversity of activities performed by an account. Specifically, BotHunter considers the total number of activities across four main groups: repository activities (forking or watching repositories), issue activities, pull request activities and commit activities.

Beyond activity counts, BotHunter incorporates temporal features that capture how frequently and how quickly actions are performed. One such feature is the median number of activities per day, which reflects the intensity of user activity. Bots tend to have significantly higher activity rates, as they can operate continuously without human limitations.

The third category, text similarity features, is designed to capture the repetitive nature of bot generated content. Since bots often automate tasks, they tend to produce highly similar or even identical messages across different interactions. To quantify this, BotHunter computes the similarity between issue and pull request comments using cosine similarity. For each account, the similarity is calculated across pairs of comments and then aggregated (using the median) across repositories. High similarity scores indicate repetitive behavior, which is a strong signal of automation.

All these features are combined and used as input to several machine learning classifiers, including Logistic Regression, K-Nearest Neighbors, Decision Tree, Support Vector Machine, and Random Forest. Among these, the Random Forest classifier achieves the best performance, with an F1-score of approximately 92.4% and an AUC exceeding 98%, indicating a high ability to distinguish between bots and humans[1]. The model is trained and evaluated on a large dataset of over 5,000 GitHub accounts, ensuring robustness and generalizability.

A key strength of BotHunter is how these features are used together rather than in isolation. The combination of profile, activity, and textual features allows the model to capture multiple aspects of behavior, avoiding relying on a single signal. For example, even if a bot does not include “bot” in its username, it can still be detected through its activity patterns or repetitive text behavior.

Overall, BotHunter represents a comprehensive and effective approach to GitHub bot detection. In the context of this thesis, BotHunter is particularly important as it provides a rich feature set that can be analyzed and ranked, offering valuable insights into which features contribute most to accurate bot detection.

Name of Feature	**Description**
Account Login	the username checked for the substring "bot"
Account Name	the display name checked for "bot"
Account Bio	short profile description checked for bot-related keywords
Account Tag	checks if user.type is equal to 'Bot'
Number of Followers	how many users follow this account
Number of Followings	how many accounts this user follows
Unique Number of Repo Activities	number of distinct types of repository actions performed
Total Number of Repo Activities	total count of repository level actions
Unique Number of Issue Activities	number of distinct types of issue actions performed
Total Number of Issue Activities	total count of issue-related actions
Unique Number of Commit Activities	number of distinct types of commit actions performed
Total Number of Commit Activities	total count of commit actions

Median Activity / Day	median number of actions performed per day
Cosine Similarity	cosine similarity of the account's own issue/PR comments
Cosine Similarity (before bot)	cosine similarity of comments made by others just before the account acted
Median Creation Time of the First Activities	Median response time to the first event in an issue
Total number of Pull Request activities	total count of PR actions
Cosine Similarity of Commit Messages	cosine similarity of commit messages

Table 2.1: Overview of Bothunter's features (reproduced from [1])

2.1.4 RABBIT

RABBIT is an open-source command-line tool designed for the efficient and scalable identification of bot accounts on GitHub [6]. It is built upon the BIMBAS classification model, which was proposed to address several limitations observed in previous bot detection approaches [5]. Unlike earlier methods that often rely on narrow feature sets or focus on a single type of activity, such as commits or comments, RABBIT adopts a broader strategy by analyzing contributor behavior through activity sequences. This allows it to detect bots across a wider range of interaction patterns.

RABBIT relies on a machine learning based binary classification model, specifically a Gradient Boosting model, trained on a wide range of features extracted from GitHub contributors. These features are primarily derived from activity sequences, which represent the ordered series of actions performed by a user over time. Instead of treating user actions as isolated events, RABBIT captures the temporal and structural relationships between these activities. This is a key aspect, as it enables the model to distinguish between human and automated behavior based on patterns rather than individual signals.

The features used in RABBIT can be broadly categorized into behavioral, temporal, and structural characteristics. One of the most important groups of features relates to the diversity and distribution of activity types. GitHub contributors can perform many different types of actions, such as creating commits, opening issues, submitting pull requests, commenting, reviewing code, and more. Human users typically exhibit a diverse range of activities, while bots are often specialized and focus on a limited subset of tasks. RABBIT captures this distinction by measuring the number of distinct activity types performed by a contributor, as well as the distribution of these activities across repositories. Features such as the total

number of activity types, the dispersion of activities per repository, and the variation in activity usage help identify whether a contributor behaves like a human or an automated system.

Another important category of features involves temporal behavior. Bots often operate with consistent, repetitive, and predictable timing patterns, while human activity tends to be more irregular. To capture this, RABBIT includes features that measure the time intervals between consecutive actions, the median and variance of these intervals, and the time taken to switch between different types of activities or repositories. For example, a bot might perform actions at very regular intervals or switch between repositories almost instantaneously, whereas a human user is more likely to show variability due to natural working patterns. These temporal features are crucial for distinguishing automation from human-driven interaction.

The design of RABBIT is therefore different from profile-oriented detectors and treats the sequence of recent activities as the main evidence for classification rather than relying mainly on explicit account metadata.

List of considered features and the intuition behind them.

Acronym	Feature	Intuition
Counting metrics:		
NA	Number of Activities	Since bots are automated agents, we expect them to produce more activities than humans as they do not suffer from the same limitations as humans (e.g., the need for sleep).
NT	Number of (activity) Types	Bots are expected to perform a smaller range of activity types than humans since they are likely to be specialised towards specific tasks.
NR	Number of Repositories contributed to	Bots are expected be involved in more repositories than humans.
NOR	Number of Owners of Repositories contributed to	Many bots are expected to be used by a small number of repository owners, as they may have been developed or configured by those owners specifically for their repositories. Humans on the other hand, have the freedom to decide which repositories they contribute, regardless of who owns the repository.
ORR	Owner-Repo Ratio = $\frac{\#NOR}{\#NR}$	We expect many bots to be used by small number of repository owners that use these bots in most of their repositories. In contrast, human accounts can be freely involved in multiple repositories belonging to a wide range of repository owners.
Aggregated metrics: Each of the metrics below is aggregated using five aggregation functions: mean, std, median, IQR, and Gini.		
NAR	Number of Activities per Repository	The number of activities per repository might be high for bots as they are intended to perform repetitive tasks and can work continually without needing breaks.
NAT	Number of Activities per Type	The number of activities per activity type might be higher for bots as they are specialised in performing specific activity types in repositories.
NCAR	Number of Consecutive Activities in a Repository	Bots do not suffer from context switching, hence they are expected to switch more easily and more frequently between different repositories.
NTR	Number of (activity) Types per Repository	We expect bots to be specialised in the activities they do within repositories, hence the number of activity types they have across repositories is more likely to be constant.
DCAR	Duration of Consecutive Activities in a Repository	If bots tend to switch between repositories, the time spent in a repository for carrying out consecutive activities might be lower than for humans.
DAAR	Duration between Activities Across Repositories	Since bots do not suffer from context switching, we expect them to take less time to have activities in multiple repositories than humans.
DCA	time Difference between Consecutive Activities	Since bots are automated scripts, they may be very fast in carrying out their next activity after the previous one. The same bot might even work in parallel in multiple, not necessarily related, repositories.
DCAT	time Difference between Consecutive Activities of different Types (in hours)	Since bots do not suffer from context switching, we expect them to switch more swiftly between activities of different types.

Table 2.2: List of features used by BIMBAS/RABBIT (reproduced from [5]).

2.2 Broader Related Work on GitHub Bots

Beyond the tools discussed above, previous research has studied software bots from several complementary perspectives. Some studies focused on textual repetition, where bots are identified because they often post similar issue comments, pull request comments, or commit messages. For example, comment-based approaches are useful for detecting bots that repeatedly post template messages in issues and pull requests, while commit-message approaches target bots that produce repetitive commit text [9], [10]. Other studies have shown that bot detection is also useful as a preprocessing step in repository analytics, because removing or separating bot activity can help avoid biased conclusions about human collaboration and contribution patterns [7].

More recent work has also examined bot behavior through GitHub activity types. Chidambaram et al. analyzed more than 830,000 activities from 713 contributors, including 305 bots and 408 humans, and considered 24 different GitHub activity types. These included actions such as opening issues, creating pull requests, commenting, reviewing code, creating tags, deleting branches, and publishing releases. Their results showed that bots usually perform fewer activity types than humans, distribute their activities more consistently across repositories, switch between repositories faster, and perform activities more regularly [3,4]. This work is important because it directly supports the activity-sequence direction later used by BIMBAS and RABBIT.

Overall, related work shows that GitHub bot detection is a multi-signal problem. Text-based methods are useful when bots leave repetitive messages, profile-based methods are useful when accounts expose bot-like names or metadata, and activity-based methods are useful when the goal is to capture behavioral patterns over time. For this reason, this thesis focuses not only on whether an account is classified correctly, but also on which feature families explain the classification and how stable the prediction is when different amounts of data are available.

2.3 Comparison of Approaches

The four bot detection approaches discussed above differ in terms of the features they use and the type of data they analyze.
BoDeGHa focuses primarily on textual similarity, making it effective for detecting bots that generate repetitive messages but less suitable for other types of bots. BIMAN extends this

approach by incorporating additional features, but it remains focused on specific activity types.

BotHunter provides a more comprehensive solution by combining multiple feature categories, including profile, activity, and text-based features. In contrast, RABBIT focuses on activity sequences and temporal patterns extracted from recent activity windows.

2.4 Summary

This section presented an overview of bot detection approaches in GitHub, focusing on BoDeGHa, BIMAN, BotHunter, and RABBIT, while also positioning them within broader related work on software bots. Each approach has strengths and limitations with respect to feature selection, data availability, scalability, and the kinds of bots it can detect.

The limitations identified in this section motivate the work presented in this thesis. In particular, this thesis aims to identify which features are most effective at detecting whether an account is a bot or a human, and to investigate how prediction confidence evolves as more activity data becomes available.

Section 3

Methodology

3.1 Overview of Methodology

The core idea is to run existing GitHub bot detection tools, export their internal feature values for a curated set of contributors, and then compare how strongly those features separate bot and human accounts.

Although the background study examined four representative approaches from literature, named BoDeGHa, BIMAN, BotHunter, and RABBIT, the experimental analysis reported here focuses on BotHunter and RABBIT. These two tools were selected because they represent two different design philosophies: BotHunter combines profile, activity, and text-similarity information, whereas RABBIT is centered on recent activity sequences and temporal behavior.

The research process consisted of four main stages. First, the ground-truth contributor list was adopted from the BIMBAS/RABBIT [5] study. Second, an additional collection pipeline was implemented to gather for each account structured JSON data through the GitHub GraphQL and REST APIs for the prefix experiment. Third, the selected tools were adapted so that they could run on the collected accounts and export their extracted features in CSV format. Finally, non-parametric statistical analysis, based on the Mann-Whitney U test and

Cliff's delta, was used to rank features according to how strongly they separate bot and human accounts.

3.2 Dataset Collection and Preparation

The dataset foundation used in this thesis was derived from the ground-truth dataset created by the developers of BIMBAS and RABBIT[5]. In their study, Chidambaram et al. constructed manually a dataset of 2,150 GitHub contributors, including 1,035 bots and 1,115 humans, all of whom were active on GitHub as of 3 May 2024. This dataset was treated in the present thesis as the source of the reference for which accounts were bots and which human. The original BIMBAS/RABBIT study created this ground-truth dataset through a multi-stage labelling process. The authors combined contributors from three prior data sources with contributors taken from the top contributors of ten popular GitHub repositories. To improve reliability, they applied an inclusion filter requiring contributors to be recently active on GitHub. They then used a multi rater manual labeling process in which two raters independently inspected each contributor's GitHub profile and activities, labeled the contributor as either bot or human, and discussed disagreements until a decision was reached. Accounts for which insufficient evidence was available were discarded. The final dataset contained both bot accounts and GitHub Apps, with the paper reporting 793 bot accounts and 242 Apps among the 1,035 bots.

Starting from this developer provided ground truth, an additional collection pipeline was built in this thesis to gather the raw GitHub data needed for the prefix experiment. The goal was not to relabel accounts, but to collect a structured per-account representation that could be reused by the selected tools and by the later statistical analysis. Each contributor was therefore stored as a separate JSON file, and for the broader collection pipeline the gathered data extended from the date an account was created up to 25 February 2026, which was the date of collection.

Data collection was performed through a combination of the GitHub GraphQL API and the GitHub REST API. GraphQL was used where flexible structured retrieval of account and repository-related information was needed, while REST endpoints were used to collect account metadata, user events, timeline evidence, and other activity records required. The resulting JSON files served as a common intermediate representation between data collection and feature extraction.

Although the original BIMBAS/RABBIT ground-truth dataset contains 2,150 contributors, this thesis does not use the full dataset for every experiment. For the feature-ranking analysis, the official test portion of the BIMBAS/RABBIT dataset was used, so that the evaluation was

performed on accounts that were not part of the model-training data. This test set contains 860 contributors, consisting of 414 bots and 446 humans [5].

From this test set, a smaller balanced subset of 200 accounts was selected for the prefix experiment. This subset contained 100 bot accounts and 100 human accounts. The reason for using this smaller subset was that the prefix experiment required collecting and processing lifetime JSON snapshots for each account, and then generating four chronological versions of every account at 25%, 50%, 75%, and 100% of its lifetime. These collected snapshots were therefore used to study how BotHunter's prediction changes when different amounts of lifetime activity data are available.

3.3 JSON Structure

Each contributor in the working dataset was represented as a separate JSON snapshot. This design allowed the collection pipeline to remain modular, reproducible, and resumable, while also making it easier to inspect the collected evidence before it was used by the selected detection tools.

The top-level structure of each JSON file contains a schema_version field, followed by the main evidence blocks: snapshot, user, repos, activity, text_samples, collection_stats, rest_evidence, and rest_user. The schema_version field was included to make the file format explicit and to support possible later changes in the collector.

The snapshot section stores metadata about the collection process, including the collection timestamp, the covered time window, the requested login, and the data source. This section is mainly used for reproducibility and traceability rather than as direct model input.

The user section stores normalized profile information about the account, such **as login, name, bio, account creation time, account update time, follower count, following count, company, location, website URL,** and **account type**. These fields are especially important for BotHunter because BotHunter makes strong use of profile-oriented features.

The repos section contains repository-level metadata associated with the account. For each repository, the JSON stores information such as the full repository name, whether it is a fork, whether it is archived or disabled, the number of stars and forks, the primary language, and repository creation, update, and push timestamps. These fields provide broader context about the repositories in which the account is active.

The activity section stores a normalized chronological view of account activity. In the JSON files, the activity items use a structure containing the activity type, repository, creation timestamp, and identifier. The main normalized activity types include commits, issues, pull

requests, issue comments, and pull-request review comments. The same section also stores daily activity counts, which are useful for computing activity volume and temporal features.
The text_samples section stores textual evidence associated with the account. It includes commit messages, issue comments, and pr review comments, together with repository names, identifiers, and timestamps. This block is especially relevant to BotHunter because BotHunter includes text similarity features.
The collection_stats section contains summary counts describing how much information was collected for the account, such as repository counts, pull request counts, issue counts, comment counts, commit-message counts, recent REST event counts, lifetime-event counts, and issue-timeline counts. In the collected snapshots, this section can also include derived BotHunter activity count fields, such as total and unique repository activities. These values are useful for data quality checking, completeness validation, and debugging the extraction pipeline.
The rest_evidence section contains raw or semi-processed evidence retrieved or reconstructed from the GitHub APIs. Besides recent REST user events, this section may include lifetime events, issue timelines, repository event items, and repository events collected through GraphQL. In some accounts, certain lists such as lifetime events or issue timelines may be empty because the corresponding evidence was not available. However, the presence of these fields keeps the structure consistent across accounts and allows each model to use the evidence it requires.
Finally, the rest_user section stores the GitHub REST user response, preserving fields such as login, numeric identifier, profile URLs, account type, public repository counts, followers, following, creation date, and update date. This block serves as both a validation layer and a fallback raw profile record, and it is more directly relevant to BotHunter than to RABBIT.

3.4 JSON Mapping

For BotHunter, the most important JSON sections were user, rest_user, text_samples, activity, and selected evidence from rest_evidence. The user and rest_user sections support profile-based features such as account login, account name, account bio, account tag, followers, and following. The text_samples section supports comment and commit-message similarity features. The activity and rest_evidence sections support activity aggregation, median activity per day, repository activity counts, and timeline-based measures. This aligns with BotHunter's overall design, which combines profile information, account activity, and textual similarity.

For RABBIT, the most important sections were activity and rest_evidence, especially the event data used to construct activity sequences. This is consistent with the BIMBAS/RABBIT approach, which is based on behavioral and temporal features extracted from contributor event histories. In other words, RABBIT depends mainly on how an account behaves over time, whereas BotHunter depends more strongly on explicit profile and textual cues.

For the prefix experiment, the activity and text_samples sections were the parts that could be sliced chronologically using their timestamps. Static profile fields such as followers, following, login, name, and bio could not be reconstructed historically for each prefix and therefore remained fixed across prefix levels. This is important for interpreting the prefix results, because BotHunter's strongest profile features were available from the earliest prefix.

Sections such as snapshot, collection_stats, schema_version, and rest_user played a supporting role in the methodology. They were particularly useful for validating the extraction process, handling interruptions, checking completeness, and maintaining reproducibility across runs. Even when they were not used directly as classification features, they were essential parts of the experimental pipeline.

3.5 Statistical Analysis

After feature extraction, the resulting datasets were analyzed to determine which features best distinguish bot accounts from human accounts. The analysis was performed separately for BotHunter and RABBIT, since the two tools rely on different feature families and different kinds of evidence. Because the data used was not normally distributed, this thesis uses non-parametric test methods called the Mann-Whitney U test, Cliff's delta, the mean and median values of each group and also accompanied by the Gini Importance measures. These measures were used together to rank features by discriminative strength, as will be described in Section 3.5.4.

3.5.1 Mann-Whitney U Test

To compare the values of each feature between bot accounts and human accounts, this thesis uses the Mann-Whitney U test. The Mann-Whitney U test is a non-parametric statistical test used to determine whether two independent groups come from the same distribution, or whether one group tends to have larger values than the other. In this study, the two groups are the set of bot accounts and the set of human accounts.

The Mann-Whitney U test does not require the data to be normally distributed, which makes it suitable for comparing feature values in this thesis. The basic idea of the test is instead of comparing raw values directly, all observations from both groups are combined and then ranked from the smallest value to the largest value. After this ranking step, the test checks whether the ranks of one group are higher or lower than the ranks of the other group. If the bot values tend to receive much higher ranks than the human values, or vice versa, then this indicates that the feature separates the two groups and have a difference between them.
The test statistic, called U, is computed from these ranks. First, the ranks of all observations belonging to one group are summed. Then, this rank sum is converted into the U statistic, which reflects how often values from one group tend to appear before values from the other group in the ordered data. A very small or very large U value suggests that the groups differ substantially, while a value near the middle suggests that the two groups overlap more strongly.
For each feature, the test produces a p-value, which indicates whether the observed difference between bots and humans is statistically significant. A small p-value means that the difference is unlikely to be due to random variation alone. However, statistical significance alone does not show how strong the difference is. For this reason, the Mann-Whitney U test was interpreted together with an effect-size measure.

3.5.2 Cliff's Delta

To measure the magnitude of the difference between bot and human accounts, this thesis uses Cliff's delta as an effect-size measure. While the p-value from the Mann-Whitney U test shows whether a difference is statistically significant, Cliff's delta shows how strong that difference is.
Cliff's delta is based on pairwise comparisons between the two groups. For every value in the bot group, it is compared with every value in the human group. The measure then evaluates how often bot values are larger than human values, and how often they are smaller. Its value ranges from -1 to 1.
A value close to 1 means that bot values are usually larger than human values. A value close to -1 means that bot values are usually smaller than human values. A value close to 0 means that the two groups overlap strongly and that the feature does not separate them well.
In this thesis, Cliff's delta is useful because it provides an interpretable measure of direction and strength. Positive values indicate that bots tend to have higher values for the feature, while negative values indicate that humans tend to have higher values. The absolute value of Cliff's delta was used to assess the strength of separation, regardless of direction. Following

the interpretation by Romano et al. [11], we consider the effect size to be negligible if $|\delta| < 0.147$, small if $0.147 \leq |\delta| < 0.33$, medium if $0.33 \leq |\delta| < 0.474$ and large if $0.474 \leq |\delta|$.

3.5.3 Descriptive Statistics

In addition to statistical significance and effect size, descriptive statistics were also computed for each feature. More specifically, the mean and median values were calculated separately for bot accounts and human accounts.

These statistics help interpret the results more clearly. For example, if a feature has a large positive effect size and also a higher bot median than human median, then it can be concluded that bots tend to show larger values for that feature. In addition, if the effect size is negative and the human median is larger, then the feature is more characteristic of human accounts.

Using descriptive statistics together with the Mann-Whitney U test and Cliff's delta makes the feature analysis easier to interpret and prevents the results from relying only on abstract statistical indicators.

3.5.4 Feature Ranking Score

After computing the p-value and Cliff's delta for each feature, the features were ranked by the absolute value of Cliff's delta in descending order. The absolute value was used because the sign of Cliff's delta only indicates direction, while the magnitude reflects how strongly the feature separates the two groups. Features with a larger absolute delta are placed higher in the ranking, meaning they are considered more discriminative between bot and human accounts.

For significance, features with $p > 0.05$ were placed at the bottom of the ranking regardless of their Cliff's delta value. This ensures that only features with statistically reliable differences between bots and humans appear at the top of the list. This approach follows the methodology used by Chidambaram et al. in the activity-type analysis that preceded BIMBAS and RABBIT, where the Mann-Whitney U test and Cliff's delta were used to identify and rank distinguishing features between bot and human contributors on GitHub [4].

3.5.5 Model-Based Feature Importance

In addition to the Mann-Whitney U test and Cliff's delta, a second independent measure of feature importance was extracted from the trained machine learning models. Both BotHunter

and RABBIT use tree-based classifiers that compute and store Gini-based feature importances during training. For BotHunter, which uses a Random Forest, and for RABBIT/BIMBAS, which uses a Gradient-boosted classifier, the feature_importances_ attribute reflects the total reduction in node impurity contributed by each feature across all trees in the model [2]. A feature with a high Gini importance was used frequently and usefully across the model's decision trees, whereas a feature with a Gini importance close to zero was rarely or never used. The Gini importances were extracted directly from the saved model files (random_forest.joblib for BotHunter and bimbas.joblib for RABBIT) and reported alongside the Cliff's delta ranking as a validation layer. When the two methods agree on the top features, it provides stronger evidence of the feature's significance. It is worth noting that Gini importance has a known limitation: it can be biased toward features with a larger number of possible values, since those features offer more potential split points. For this reason, Gini importance is not used as the primary ranking criterion but as a complementary measure to validate the order produced by Cliff's delta.

Finally, all features were sorted in descending order of their absolute Cliff's delta value, with non-significant features ranked last. The resulting ranking is reported in Section 4 and used to compare the most discriminative features of BotHunter and RABBIT.

Section 4

Ranking Results

4.1 Feature Importance Analysis

This chapter presents the results of the feature ranking analysis for BotHunter and RABBIT. The objective is not only to report which features ranked highest, but also to interpret what these rankings reveal about the kinds of evidence each model uses to distinguish bot accounts from human accounts.

For each numeric feature, the Mann-Whitney U test was used to compare the bot and human distributions. In addition to the p-value produced by the Mann-Whitney U test, Cliff's delta was computed to estimate the practical size of the difference between the two groups. Following the interpretation thresholds used by Romano et al. and by the activity-type study that preceded BIMBAS, the effect size is treated as negligible when $|\delta| < 0.147$, small when $0.147 \leq |\delta| < 0.33$, medium when $0.33 \leq |\delta| < 0.474$, and large when $|\delta| \geq 0.474$ [4, 11]. Mean and median values were also reported for the bot and human groups so that the direction of the separation could be interpreted. Features were then ranked by the absolute value of Cliff's delta in descending order, with features that did not reach statistical significance ($p > 0.05$) placed at the bottom of the ranking regardless of their effect size. As a complementary

measure, Gini-based feature importances were extracted directly from the trained model files and reported alongside the Cliff's delta ranking as a validation method.

4.2 BotHunter Feature Importance

The results show that although BotHunter combines profile, activity, and text-similarity features, its strongest discriminative signals in this dataset are profile-oriented features, like account login, followers, following, and account name. The features separate into three distinct tiers based on the strength of their discriminative significance.

4.2.1 Top Tier: Profile Metadata

The four strongest BotHunter features are Account login, Number of followers, Number of following, and Account name. These features all carry large effect sizes and represent the strongest signals in Bothunter’s classification.

Account login is the single strongest feature in both the statistical and model based rankings, with a Cliff's delta of 0.785 and a Gini importance of 32.6%. BotHunter applies a regular expression that fires when the username ends in "bot" and in the RABBIT ground truth dataset the vast majority of bot accounts carry this pattern explicitly. A delta of 0.785 means there is an 87% probability that a randomly chosen bot scores higher than a randomly chosen human on this feature. Account login alone accounts for more than a third of the model's total classification improvement across all trees, making it an extremely strong signal though one with a clear limitation: any bot that uses a human-sounding username would score zero on this feature, contributing nothing to the classification decision.

Number of followers ranks second statistically ($\delta = 0.744$), with bots showing a median of 3 followers compared to a human median of 82. Human developers build reputations over time, get followed by collaborators and project users, and participate actively in the GitHub social network. Bots are rarely followed by anyone except accounts that interact with the repositories they serve. Its Gini rank of 5th is slightly lower than the statistical rank of 2nd, suggesting the model does not rely on follower count quite as much as the delta implies when other features are also available.

Number of following ranks third ($\delta = 0.694$), with bots showing a median of 0 compared to a human median of 6. Bots almost never follow other accounts since there is no reason for them to do so. The Gini rank of 3rd, higher than the statistical rank, indicates the model weighs this feature heavily in combination with Account login and Account name to build a strong profile-based signature.

Account name ranks fourth (δ = 0.535), with BotHunter applying the same bot/automate pattern to the account's display name. The lower delta compared to Account login reflects that fewer bots carry the bot pattern in their display name than in their username. The Gini rank of 2nd at 14.5% importance is notably high relative to its statistical rank, suggesting the model finds the name pattern particularly useful when combined with other features.

4.2.2 Middle Tier: Activity Diversity and Volume

Activity related features contribute meaningfully to the ranking while carrying medium to small effect sizes.

Unique number of Repo activities (δ = 0.360) measures the diversity of repository level action types; humans perform a wider variety of these actions because they interact with repositories in multiple roles, whereas most bots perform at most one type of repository-level action.

Total number of Repo activities (δ = 0.208) tells the same directional story but with a smaller effect, suggesting that the diversity of repo actions matters more than raw volume.

Unique number of Issue activities (δ = 0.206) and Median Activity per Day (δ = 0.151) complete this tier.

4.2.3 Bottom Tier: Negligible and Non-Significant Features

The remaining features carry negligible effect sizes or do not reach statistical significance. Several of these reflect genuine data availability limitations rather than an absence of discriminative power.

Unique number of Commit activities ranks 9th (δ = 0.129, Gini rank 17th) and the negligible effect size means this feature offers very little practical discriminative power between bots and humans on its own.

Cosine similarity ranks 10th (δ = 0.099, Gini rank 4th). This is a notable disagreement between the statistical and model-based rankings. The Gini rank of 4th suggests the model still uses this feature frequently during tree construction, yet the low Cliff's delta indicates it has little practical ability to separate bot and human distributions in the dataset.

Cosine similarity of Comments Before Bot ranks 11th (δ = 0.091,Gini rank 15th) and Total number of Commit activities ranks 12th (δ = 0.087, negligible,Gini rank 10th), with bots showing a median of 3 and humans a median of 9, meaning humans make slightly more commits on aggregate in this dataset.

Account bio ranks 13th (δ = 0.022,Gini rank 18th), with both groups showing a median of 0. Despite statistical significance, the effect is essentially zero. Total number of Issue activities ranks 14th (δ = 0.053, Gini rank 8th), again with both medians at 0.

Account tag (δ = 0.000) checks whether GitHub's REST API returns type="Bot". In the RABBIT ground truth, most bot accounts are not officially flagged by GitHub as Bot type because GitHub only assigns this type to accounts authenticating as GitHub Apps. Many script-based bots authenticate as regular users and are never flagged, rendering this feature uninformative in this dataset despite it theoretically being the most direct signal available. The remaining features in the bottom tier show similarly negligible or non-significant results. Overall, the BotHunter ranking confirms that the model relies primarily on visible metadata and profile cues, while activity signals play a more supporting role in this dataset.

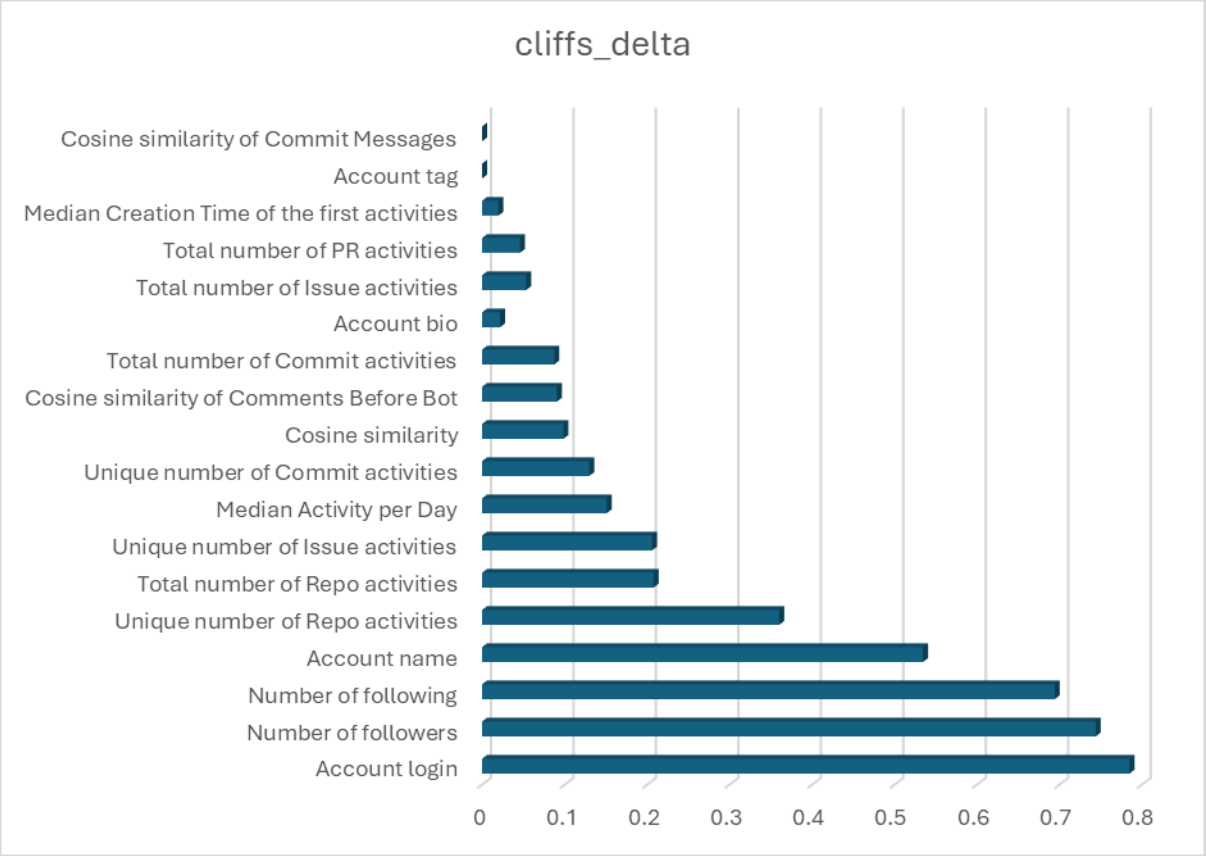

Figure 4.1 Feature importance ranking for BotHunter based on Cliff's delta

4.3 RABBIT Feature Importance

The RABBIT results show a different ranking pattern which is driven by behavioral and temporal patterns derived from recent activity sequences, reflecting a fundamentally different detection method.

4.3.1 Top Tier: Activity Diversity and Temporal Regularity

The eight strongest RABBIT features all carry large effect sizes and collectively describe bots as more specialized and active than humans.

NT (Number of Activity Types) is the single most discriminative feature in both the statistical and model based rankings ($\delta = 0.713$, Gini importance 19.4%). Bots show a median of 3 activity types compared to a human median of 8. Bots are specialists implying that they perform one or two specific task types consistently while humans interact across many activities like pushing commits, opening issues, commenting on pull requests, creating releases, reviewing code, and more.

NOR (Number of Repositories) ranks second ($\delta = 0.614$), with bots showing a median of 1 repository compared to a human median of 3. Humans contribute to a wider range of repositories from different owners and organizations, whereas bots are typically deployed within a single organization or a small cluster of related repositories.

NAT_median (Median Time Between Consecutive Activities) ranks third ($\delta = 0.611$), with bots showing a median gap of 15.5 hours compared to 4.0 hours for humans. The Gini rank of 3rd at 10.8% confirms the model treats this as one of its most important features.

NTR_gini (Gini Coefficient of Activity Types per Repository) ranks fourth ($\delta = 0.608$). A higher Gini means more unequal distribution of activity types across repositories. Humans perform a more unequal mix heavily active in some repositories and barely active in others whereas bots distribute their narrow set of activity types more evenly across all repositories.

NTR_std (Standard Deviation of Activity Types per Repository) ($\delta = 0.582$) captures the same pattern through variance: humans show more variability in the number of activity types they perform per repository, while bots consistently perform the same actions everywhere.

NAT_mean (Mean Time Between Consecutive Activities) ranks sixth statistically ($\delta = 0.559$) but holds 2nd place in Gini importance at 16.1%.

DCAT_std (Time Difference between Consecutive Activities of Different Types) ($\delta = 0.549$) and DAAR_std (Duration between Activities Across Repositories) ($\delta = 0.519$) complete the top tier, both measuring the standard deviation of timing between activity type switches and repository switches respectively. In both cases humans show far greater variability.

4.3.2 Middle Tier

NAR_median (δ = 0.474) shows bots concentrating more activity per repository (median 9.5 and 3.0 for humans), consistent with the picture of bots being deployed in a small number of repositories where they act heavily.
DCAT_mean and DAAR_mean (both around δ = 0.413) capture mean switching times between activity types and repositories respectively, with humans taking longer on average reflecting the human tendency to focus on one type of task at a time.
One notable disagreement appears with DCAT_median (δ = 0.373, statistical rank 16th, Gini rank 4th). Despite its moderate statistical ranking, the model treats it as the 4th most important feature overall.
The remaining features in this tier including DAAR_IQR, DCAT_IQR, and NAR_gini all measure dispersion in timing or activity distribution from slightly different perspectives.

4.3.3 Bottom Tier: Negligible Features

The bottom tier is essentially uninformative in this dataset.
NA (Number of Activities) (δ = 0.133) shows bots with slightly more total activities than humans (median 89 vs 67).
ORR (Owner-to-Repository Ratio) ranks 33rd (δ = 0.118, Human > Bot). This feature measures the ratio of distinct repository owners to the total number of repositories in which a contributor is active. The direction is Human > Bot, with a bot median of 0.5 compared to a human median of 0.6, suggesting humans interact across a slightly broader range of repository owners. However, the negligible effect size makes this feature practically useless for classification on its own.
NTR_median (δ = 0.020) is the weakest feature in the entire RABBIT ranking, with both groups showing an identical median of 2.0 activity types per repository.
The remaining 20 features in the bottom tier including NAT_IQR, NAT_std, DCAR_mean, NAR_IQR, DCAR_median, DCA_gini, NCAR_mean, DCA_median, NCAR_IQR, NCAR_std, DCA_mean, DCAR_IQR, DCAR_std, DCAT_gini, NTR_mean, NAR_mean, DAAR_gini, DCA_std, DAAR_median, and NAR_gini all fall in the small or negligible range. None of these adds meaningful discriminative power beyond what the top and middle tiers already provide.

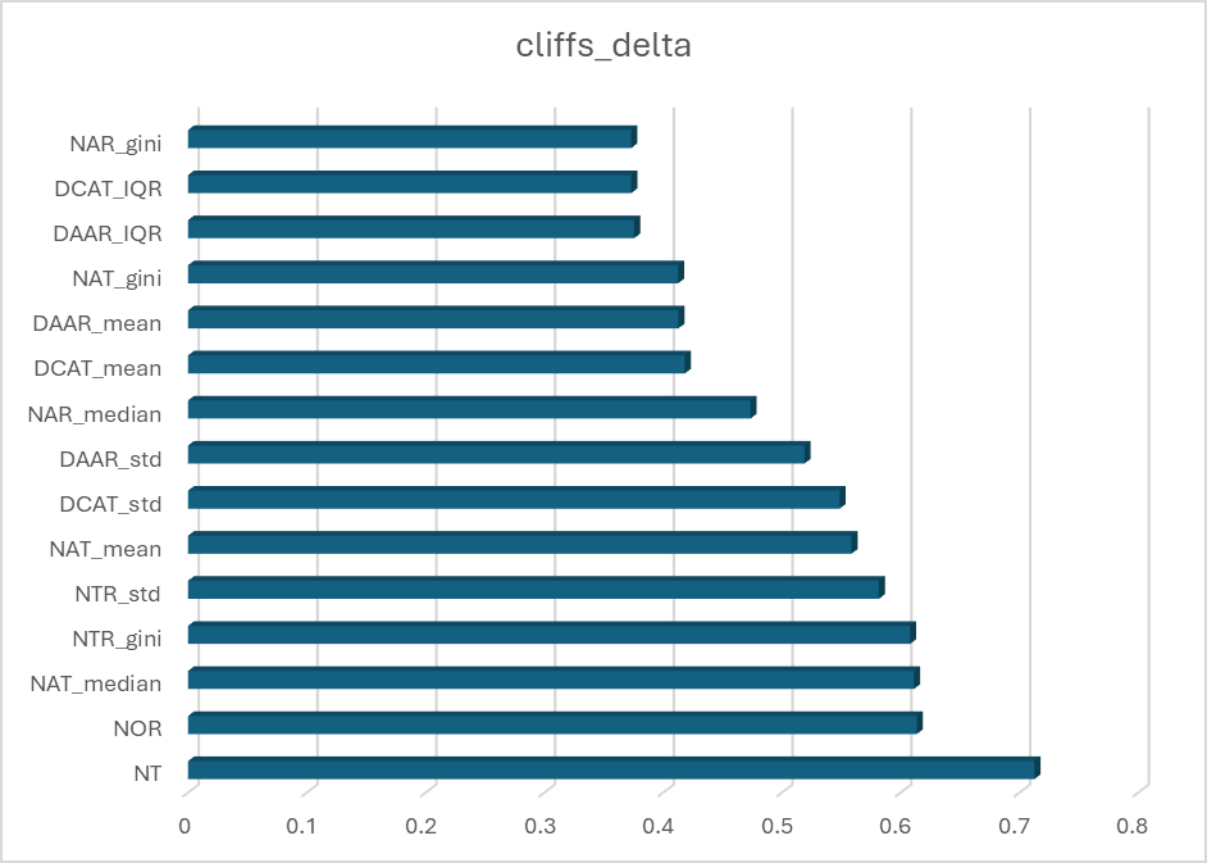

Table 4.2 Feature importance ranking for RABBIT based on Cliff's delta

4.4 Comparison of Models

Taken together, the two rankings reveal a clear contrast between BotHunter and RABBIT. Although both tools are designed to distinguish bots from humans on GitHub, they succeed by exploiting very different kinds of evidence.

BotHunter tends to lean more on profile oriented features such as account login, account name, number of followers, and number of following. In other words, BotHunter exploits what the account looks like. Although the presence of the substring "bot" in a login or display name, combined with a weak social footprint are enough to drive classification to a high confidence, having a high social footprint could lead to the opposite.

RABBIT, by contrast, is dominated by recent behavioral and temporal features: the number and diversity of activity types, the number of repositories, and a collection of timing statistics describing how regularly and predictably an account operates. It mainly exploits how the account behaves within a recent event history. This approach remains informative even when bots do not explicitly identify themselves, though it depends on a sufficient amount of recent events to compute reliable feature values.

4.5 Summary of Findings

The results of this study can be summarized in three main findings. First, BotHunter places at the top of the ranking profile-oriented features, like Account login ($\delta = 0.785$), Number of followers ($\delta = 0.744$), Number of following ($\delta = 0.694$), and Account name ($\delta = 0.535$). Second, RABBIT is dominated by behavioral and temporal activity sequence features, with NT ($\delta = 0.713$), NOR ($\delta = 0.614$), NAT_median ($\delta = 0.611$), NTR_gini ($\delta = 0.608$), and NAT_mean ($\delta = 0.559$) leading the ranking. Third, the comparison shows that the two models rely on fundamentally different evidence: BotHunter mainly uses visible profile cues, whereas RABBIT mainly uses recent behavioral signatures.

Section 5

Prefix Experiment Results

5.1 Overview

The feature ranking analysis in Section 4 showed that BotHunter relies for the most part on profile-oriented signals, such as the account login name. This raises an important follow-up question: how much of an account's lifetime data does BotHunter actually need to make a reliable prediction, and does its classification improve when more behavioral evidence becomes available over time?

To investigate this, a prefix experiment was designed and conducted exclusively for BotHunter. The experiment runs BotHunter on chronological slices of each account's lifetime data at 25%, 50%, 75%, and 100% of the time between account creation and the date of data collection. For each prefix level, the classification and the bot probability score are recorded. By comparing the four prefix levels, it is possible to see whether BotHunter detects bots early or whether the model needs a longer activity history to reach a reliable prediction.

RABBIT was not included in this experiment because it is constrained to recent GitHub event history and, in the implementation used in this thesis, retrieves a limited recent event window. The prefix experiment is therefore specific to BotHunter, because the data collection pipeline built for this thesis allows full lifetime histories to be sliced chronologically and supplied to the model as input.

5.2 Experimental Design

The experiment was conducted on a balanced sample of 200 accounts drawn from the RABBIT/BIMBAS test set: 100 human accounts and 100 bot accounts. The bot sample was

further divided into two groups based on whether the account activates BotHunter's Account login feature. This split allows the experiment to evaluate separately the bots that self identify through naming and the bots that do not carry an explicit naming signal.

The human accounts were selected to represent a range of activity levels, including accounts with high, medium, and low activity. This was done to avoid bias in the results toward only very active or very inactive users. The final comparison file contains one prediction for every account at every prefix level, so the final analysis is based on the complete set of 200 accounts.

For each account, lifetime data was collected from account creation up to the collection date in 25th February 2026. The lifetime was divided into four chronological prefix windows: the first 25%, the first 50%, the first 75%, and the full 100%. BotHunter was run separately on each version, with its standard GitHub API calls replaced by the pre-collected JSON data. The Random Forest model was also modified to output the bot probability using predict_proba, so that changes in confidence could be observed in addition to changes in the final binary prediction.

The event history was sliced chronologically, but some profile and social features could not be reconstructed historically from the collected GitHub data. In particular, the number of followers and the number of following, as well as profile fields such as account name, bio, and account type, were available only as values from the collected snapshot rather than as values at each prefix cutoff. These features were therefore kept constant across prefix levels. As a result, the experiment mainly tests how BotHunter changes when activity evidence is reduced, while the strongest profile features remain available at all prefixes.

5.3 Results

The final prefix comparison was performed on the complete set of 200 accounts, consisting of 100 bot accounts and 100 human accounts. Unlike the earlier partial runs, the final comparison file contains predictions for all four prefix levels: 25%, 50%, 75%, and 100% of the available lifetime data. In this experiment, the 100% prefix is equivalent to the full lifetime result.

Prefix level	Correct	Wrong	Accuracy	Correct bots	Missed bots	Correct humans	False positives
25%	162 / 200	38 / 200	81.0%	66	34	96	4
50%	162 / 200	38 / 200	81.0%	66	34	96	4
75%	166 / 200	34 / 200	83.0%	70	30	96	4
100% / Lifetime	165 / 200	35 / 200	82.5%	69	31	96	4

Table 5.1: Prefix experiment results for BotHunter.

The results show that increasing the amount of lifetime data does not lead to a strong improvement in BotHunter performance. Accuracy is 81.0% at both the 25% and 50% prefixes, increases to 83.0% at the 75% prefix, and then slightly decreases to 82.5% at the full lifetime level. This means that the prediction does not substantially improve as more data is added.

The main imbalance is between bot recall and human accuracy. Human accounts are identified very consistently: 96 out of 100 humans are correctly classified at every prefix level. BotHunter detects 66 bots at 25% and 50%, 70 bots at 75%, and 69 bots at full lifetime. Therefore, the main source of error is not false positives but false negatives, meaning real bot accounts predicted as human.

The number of available events increases substantially across the prefix levels. The median number of events rises from 903 events at the 25% prefix to 1,807 at 50%, 2,766 at 75%, and 3,611 at the full lifetime level. However, this increase in available activity data does not produce a proportional increase in accuracy. This suggests that BotHunter is not limited only by the amount of data, but by the type of signal present in the account.

This limitation helps explain why the accuracy changes only slightly as the prefix grows. Because profile features such as Number of followers, and Number of following are constant across prefixes, the experiment is not a perfect historical reconstruction of what BotHunter would have seen at the 25%, 50%, or 75% points of an account's life. Instead, it is a partial-activity experiment: the activity history is reduced chronologically, but profile metadata remains fixed. This may make the early prefixes appear stronger than they would have been if historical follower and following values had been available.

The prediction changes between prefix levels are limited. From 25% to 50%, only two accounts change prediction: sul-devops-team changes from Bot to Human, while for example miss-islington changes from Human to Bot. These changes cancel each other out, so the total accuracy remains the same. From 50% to 75%, four bot accounts become correctly detected: genie-jnosql, JuliaRegistrator, kie-ci3, and SonataCI. This explains why the 75% prefix gives the best result. From 75% to full lifetime, kie-ci3 changes back from Bot to Human, which explains the small drop in the final result.

The false positives are also stable. The same four human accounts are classified as bots at every prefix level: brunosabot, jasoncabot, russellbot, and UEWBot. These accounts are labeled as humans in the ground truth, but their usernames contain bot-like strings.

5.4 Named vs Unnamed Bot Analysis

To better understand the prefix results, the bot accounts were divided into two groups. The first group contains bots that activate BotHunter's Account login feature, meaning that their username contains a bot-like pattern. The second group contains bots that do not activate this naming feature. This separation is important because the feature ranking analysis in Section 4 identified Account login as the strongest BotHunter feature.

Prefix level	Named bots detected	Named bot recall	Unnamed bots detected	Unnamed bot recall
25%	47 / 47	100.0%	19 / 53	35.8%
50%	47 / 47	100.0%	19 / 53	35.8%
75%	47 / 47	100.0%	23 / 53	43.4%
100% / Lifetime	47 / 47	100.0%	22 / 53	41.5%

Table 5.2: Detection of named and unnamed bots across prefix levels.

The difference between the two groups is clear. All 47 named bots are detected correctly at every prefix level, giving 100% recall for this group. In practice, BotHunter does not need much behavioral evidence to detect these accounts, because the username signal is already strong enough from the earliest prefix.

The unnamed bots are much harder to detect. At the 25% and 50% prefixes, only 19 out of 53 unnamed bots are correctly classified. At 75%, this improves to 23 out of 53, and at full lifetime it becomes 22 out of 53. Even with the complete lifetime history, most unnamed bots are still classified as human.

The bot probability values support this interpretation. For named bots, the mean bot probability remains high and stable across all prefix levels, around 0.78. For unnamed bots, the mean bot probability increases gradually from about 0.43 at 25% to about 0.47 at full lifetime, but it remains below the 0.50 classification threshold on average. Additional activity data therefore makes some unnamed bots appear slightly more bot-like, but not enough to change the prediction for most of them.

These results confirm that BotHunter performance is largely determined by whether the bot exposes an explicit profile-level naming signal. When that signal exists, the tool detects the account immediately. When it is absent, additional lifetime data helps only slightly.

5.5 Summary

The prefix experiment provides three main findings. First, BotHunter performs reasonably well with only 25% of the account lifetime, reaching 81.0% accuracy. However, this

performance does not improve substantially as more data is added. The best result is 83.0% at the 75% prefix, while the full lifetime result is 82.5%.

Second, BotHunter is much better at identifying human accounts than bot accounts. Human accuracy remains stable at 96% across all prefix levels, while bot recall remains between 66% and 70%. This shows that the main weakness of BotHunter in this experiment is the number of bot accounts that are classified as human.

Third, the experiment confirms that the presence of a naming signal is a strong factor affecting BotHunter's prediction. Named bots are detected perfectly at all prefix levels, while unnamed bots are mostly missed even when full lifetime data is available. Therefore, for BotHunter, more data does not necessarily provide a better prediction. The type of evidence matters more than the quantity of evidence.

Finally, the results should be interpreted with the profile-feature limitation in mind. Since follower and following counts could not be derived separately for each prefix, the experiment measures the effect of reducing activity evidence rather than the effect of reducing every feature category equally. Despite this limitation, the prefix experiment complements the Ranking experiment where we can observe that even with large amounts of activity if the profile metadata are kept the same the prediction remains closely the same throughout all prefixes, indicating that profile metadata play an important role in Bothunters classification.

Section 6

Discussion

6.1 Contributions

This thesis makes three main contributions to the study of GitHub bot detection. First, it provides an empirical comparative analysis of feature importance for two tools with different design philosophies. This shows that the most informative features depend strongly on the type of evidence the detector was designed to exploit.
Second, the prefix experiment provides evidence on how BotHunter's prediction evolves across an account's lifetime. The results show that performance is largely determined at the beginning by profile cues rather than progressively improving through accumulated behavioral evidence. This is useful for practitioners who want to understand whether a detector can identify automation early.
Third, the thesis demonstrates that profile-based and behavior-based approaches are complementary. BotHunter is highly effective when bots self-identify through naming, while RABBIT focuses on activity-sequence behavior. This suggests that a combined approach using both profile and behavior signals could be stronger than either approach alone, especially for unnamed bots.

6.2 Implications of the Results

The results have implications for several groups. Mining software repositories researchers can use these findings when choosing a bot detection tool as a preprocessing step. If bot accounts are not detected correctly, automated activity may be incorrectly treated as human

behavior, which can bias studies of productivity, collaboration, review practices, and contribution patterns.

The results are also useful for repository maintainers and open-source project administrators. Maintainers often need to understand which accounts interacting with their repositories are humans and which are automated systems. The findings show that accounts with obvious bot-like names are easy to detect, but accounts that behave like automation while using human-like profiles may remain hidden.

Tool developers can also benefit from the findings. The comparison between BotHunter and RABBIT shows that profile-based and behavior-based features should not be treated as competing alternatives, but as complementary evidence. A future detector could combine account metadata, social features, activity diversity, timing patterns, and textual behavior to reduce the weaknesses of each individual approach.

Finally, the prefix experiment has implications for early bot detection. For named bots, BotHunter can make accurate predictions with little lifetime data. For unnamed bots, however, more data only provides a small improvement. This means that early bot detection is possible for obvious automation, but remains difficult for accounts that are configured to look like ordinary users.

6.3 Limitations

Several limitations of this work should be acknowledged. Reporting these limitations is important for clarifying the validity of the experimental conclusions. The most significant limitation is that the prefix experiment was conducted only for BotHunter. RABBIT was not included because the version used in this thesis relies on recent GitHub event history and is limited by the GitHub Events API, which prevent it from getting the lifetime data it needs to make a fully complete prediction by using all of it’s features.

A second limitation concerns historical feature reconstruction in the prefix experiment. The activity evidence was sliced chronologically, but historical values for some profile and social features were not available. In particular, followers and following counts could not be derived separately for the 25%, 50%, and 75% prefixes. These values were therefore taken from the collected account snapshot and reused across prefix levels. This affects the interpretation of the results because several of BotHunter's strongest features remain constant even at early prefixes.

A third limitation concerns bot subgroups.It is possible that the most informative features differ between different kinds of bots like dependency bots, CI bots, moderation bots, release bots, and GitHub Apps.

Section 7

Conclusion and Future Work

7.1 Summary

This thesis investigated feature importance in GitHub bot detection through a comparative analysis of two detection tools: BotHunter and RABBIT. The motivation for this work comes from the growing presence of automated accounts on GitHub and the methodological problems this creates for software engineering research. When bot activity is not identified and separated from human activity, analyses of productivity, collaboration, and contribution patterns can become biased.

The study used the ground-truth dataset constructed by Chidambaram et al.[5] for the BIMBAS/RABBIT study. Both tools were run on the selected test-set accounts, and the extracted feature values were analyzed using the Mann-Whitney U test and Cliff's delta as the primary statistical measures. Gini-based feature importances extracted from the pre-trained model files were used as a complementary validation layer. A prefix experiment was then conducted for BotHunter using 200 accounts and four chronological slices of lifetime data: 25%, 50%, 75%, and 100%.

The results show that BotHunter and RABBIT rely on different kinds of evidence. BotHunter is mainly driven by profile-oriented features, especially account login and social metadata, while RABBIT is mainly driven by behavioral and temporal activity-sequence features. The prefix experiment further shows that BotHunter does not significantly improve when more lifetime data is added, because the strongest signals are often already present early in the account profile.

7.2 Answers to Research Questions

RQ1: Which features are the strongest indicators for distinguishing GitHub bot accounts from GitHub human accounts?

For BotHunter, the strongest indicators are profile-oriented features. Account login is the most discriminative feature ($\delta = 0.785$), driven by the presence of the substring "bot" at the end of the username. Number of followers ($\delta = 0.744$) and Number of following ($\delta = 0.694$) follow closely, reflecting the weaker social footprint of bot accounts compared to human developers. Account name ($\delta = 0.535$) completes the top profile tier. Activity-related features contribute, but with lower effect sizes.

For RABBIT, the strongest indicators are behavioral and temporal features derived from recent activity sequences. The Number of Activity Types (NT, $\delta = 0.713$) is the most discriminative feature, reflecting the specialist nature of many bot accounts. The Number of Repositories (NOR, $\delta = 0.614$), NAT_median, NTR_gini, NTR_std, NAT_mean, DCAT_std, and DAAR_std form a coherent top tier that describes bots as more temporally regular, more activity-concentrated, and less diverse than humans.

RQ2: How much data does a model need in order to provide a better prediction, and how does the prediction change when less or more data is available?

The prefix experiment shows that BotHunter does not substantially benefit from additional lifetime data. Accuracy is 81.0% at 25%, 81.0% at 50%, 83.0% at 75%, and 82.5% at full lifetime. The best result appears at 75%, not at 100%, which means that the prediction is not strictly monotonic. More data can help some accounts, but it can also reduce the bot probability of others.

The main explanation is the difference between named and unnamed bots. All 47 named bots are detected correctly at every prefix level. In contrast, only 22 out of 53 unnamed bots are detected at full lifetime. Therefore, for BotHunter, the type of evidence matters more than the quantity of evidence. If the account contains a strong naming signal, little data is needed. If the naming signal is absent, even full lifetime data may not be enough.

This answer should be interpreted with one important qualification. Profile and social features, like followers and following, could not be reconstructed historically for each prefix. Therefore, the experiment shows how predictions change when lifetime activity is sliced, but not how predictions would change if every profile feature were also available historically. However, this indicates how much two features like number of followers and number of followings can affect the prediction even though the model takes as input a lifetime’s worth of activity.

7.3 Future Work

Several directions exist for extending this work. The most natural next step is to conduct a full lifetime prefix experiment for a behavior-based model. If the data limitations of RABBIT can be addressed through sources such as GH Archive or World of Code, a similar prefix analysis could compare how profile-based and behavior-based evidence accumulate across account lifetime.

A second direction is category-specific analysis. The current study treats all bots as a single group, but different bot categories may behave differently. Repeating the feature ranking analysis separately for dependency bots, CI bots, release bots, moderation bots, and GitHub Apps would provide more targeted guidance for practitioners.

A third direction is the development of a combined model. The results of this thesis suggest that a tool combining BotHunter's profile signals with RABBIT's activity-sequence signals could outperform either approach alone, especially for unnamed bots where BotHunter currently struggles.

Bibliography

[1] Abdellatif, A., Wessel, M., Steinmacher, I., Gerosa, M. A., & Shihab, E. (2022). BotHunter: An Approach to Detect Software Bots in GitHub. In Proceedings of the 19th International Conference on Mining Software Repositories (MSR 2022), 15-26. ACM. https://doi.org/10.1145/3524842.3527959

[2] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

[3] Chidambaram, N., Decan, A., & Mens, T. (2023a). A Dataset of Bot and Human Activities in GitHub. In Proceedings of the 20th International Conference on Mining Software Repositories (MSR 2023), 465-469. IEEE. https://doi.org/10.1109/MSR59073.2023.00070

[4] Chidambaram, N., Decan, A., & Mens, T. (2023b). Distinguishing Bots From Human Developers Based on Their GitHub Activity Types. CEUR Workshop Proceedings, 3483, 31-39. https://ceur-ws.org/Vol-3483/paper3.pdf

[5] Chidambaram, N., Decan, A., & Mens, T. (2025). A bot identification model and tool based on GitHub activity sequences. Journal of Systems and Software, 221, Article 112287. https://doi.org/10.1016/j.jss.2024.112287

[6] Chidambaram, N., Mens, T., & Decan, A. (2024). RABBIT: A tool for identifying bot accounts based on their recent GitHub event history. In Proceedings of the 21st International Conference on Mining Software Repositories (MSR 2024). ACM. https://doi.org/10.1145/3643991.3644877

[7] Golzadeh et al. (2022), "Recognizing bot activity in collaborative software development," IEEE Software, 39(5), 56–61 (doi: 10.1109/ms.2022.3178601)

[8] Dey, T., Mousavi, S., Ponce, E., Fry, T., Vasilescu, B., Filippova, A., & Mockus, A. (2020). Detecting and Characterizing Bots that Commit Code. In Proceedings of the 17th International Conference on Mining Software Repositories (MSR 2020), 209-219. ACM. https://doi.org/10.1145/3379597.3387478

[9] Golzadeh, M., Decan, A., & Mens, T. (2020). Evaluating a bot detection model on git commit messages. In 2nd International Workshop on Bots in Software Engineering (BotSE 2020), CEUR Workshop Proceedings, 2912.

[10] Golzadeh, M., Decan, A., Legay, D., & Mens, T. (2021). A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. Journal of Systems and Software, 175, Article 110911. https://doi.org/10.1016/j.jss.2021.110911

[11] Romano, J., Kromrey, J. D., Coraggio, J., Skowronek, J., & Levine, L. (2006). Exploring methods for evaluating group differences on the NSSE and other surveys: Are the t-test and Cohen's d indices the most appropriate choices? Paper presented at the Annual Meeting of the Southern Association for Institutional Research.

[12] Wessel, M., de Souza, B. M., Steinmacher, I., Wiese, I. S., Polato, I., Chaves, A. P., & Gerosa, M. A. (2018). The Power of Bots: Characterizing and Understanding Bots in OSS Projects. Proceedings of the ACM on Human-Computer Interaction, 2(CSCW), Article 182. https://doi.org/10.1145/3274451