A Survey of Recent Artificial Intelligence-Driven Frameworks for User-Level Activity Prediction in Github

Fred Mubang[a,*], Lawrence O. Hall[a]

[a]*Computer Science, University of South Florida, 4202 E Fowler Ave, TAMPA, 33613, Florida, United States*

Abstract

This work is a survey of recent Artificial Intelligence driven approaches of user-level activity prediction in the online collaborative platform, Github. This work is motivated by the fact that Github is used by over 73 million developers, 4 million organizations, and has over 200 million code repositories. Futhermore, 84% of the Fortune 100 companies utilize Github for their software development. By understanding how to leverage AI to predict user activity in Github, one could potentially predict future technological trends and detect future malware attacks. A myriad of AI-driven approaches are discussed in this work, such as various neural network-based approaches, clustering based approaches, and several others.

Keywords: artificial intelligence, machine learning, network science, graph prediction

1. Introduction

In recent times, the online collaborative platform, Github was grown strongly in popularity. As of the time of this writing, Github is used by over 73 million developers, 4 million organizations, and has over 200 million code repositories. Also, 84% of the Fortune 100 companies utilize Github for their software devel-

*Corresponding author.
Email addresses: `fmubang@usf.edu` (Fred Mubang), `lohall@usf.edu` (Lawrence O. Hall)

opment [1]. As one can see, Github is a ubiquitous part of the technology all around us.

Furthermore, in recent times, there has been increased interest in building frameworks that can predict future user activity in Github. Since Github is so widely used, if one could predict future user activity on this platform, such predictions could make one aware of future technological advances, or alert one to any potential cybersecurity threats.

Many of the recent Github prediction frameworks heavily utilize cutting-edge Artificial Intelligence approaches, such as Long Short Term Neural Networks in the cases of [2, 3], or Diffusion Recurrent Convolutional Neural Networks in the case of [4]. With the increase in AI prediction frameworks, the task of selecting the most appropriate one might be daunting to someone new to the field. With all this in mind, we present this survey on recent AI approaches for Github user prediction. Specifically, this paper makes the following contributions:

1. Recent artificial intelligence-driven frameworks within the domain of Github user activity prediction are discussed.
2. The performance results of each framework presented are discussed.
3. Potential real-world applications are discussed for each framework.
4. Potential future work directions for each framework are discussed.

This paper is organized as follows. Section 2 contains the motivation for predicting user activity in Github. Section 3 contains the definitions for the terms used throughout this work. Section 4 discusses the challenges of Github user prediction. Section 5 discusses the different prediction frameworks covered in our literature review. In Section 6 the results of each framework are discussed. Section 7 discusses potential real-world applications. Section 8 discusses directions of future work and lastly Section 9 contains an overall summary.

2. Motivation for Predicting Github User Activity

Since Github is so widely used, if one can predict future user and repository activity, one can gain a better understanding of technological advances. This information might be useful for companies trying to learn how to better service their customers with software that is relevant to their needs. For example, if one had a model that could predict that there would be an increase in activity on repos related to Linux in the future, companies that use Linux-based products could focus their efforts on improving the quality of their products, or adding new features to them in order to attract new customers.

Furthermore, predicting Github activity can aid with cybersecurity maintenance. In 2016, Black Duck's Center for Open Source Research and Innovation (COSRI) analyzed more than 1,000 applications that were audited as part of merger-and-acquisition transactions [5]. The audit analysis found that 96% of these applications contained open-source software, and more than 60% of those applications contained known open-source security vulnerabilities [5]. If an organization could predict that repositories containing vulnerable open-source code will receive a higher-than-normal amount of activity, that could alert an organization to the fact that their systems utilizing software from those repositories could be the target of hackers. The organization could then take preemptive measures to ensure any private data they have will be safe. The authors of [3] found that software vulnerabilities are mentioned on Reddit and Twitter, and that this information can be used to predict repository activity on Github. So, it is possible to use data-driven models to avoid or prevent software vulnerability exploitation.

Such foresight may have been useful to Equifax, a credit bureau, who in 2017 was the victim of a data breach in which the social security numbers of 143 million Americans were put at risk [5]. The hackers were able to attack Equifax due to a software vulnerability in Apache Struts, an open-source framework (on Github) for creating web applications in Java. The specific identifier for this vulnerability is *CVE-2017-5638* [5].

3. Temporal Graph Definitions and Measurement Discussion

3.1. Temporal Graphs

The works covered in this survey predict activity in Github temporal graphs. In Github, the temporal graph is represented as the tuple $G = (U, R, E, T)$. U is the set of all users such that $u \in U$ represents a user. R is the set of repos such that $r \in R$ represents a repo. The set, E can be thought of as a set of edge tuples, each of which having the form (u, r, a, t, w). This represents the interaction between user u and repo r. The term, a represents an action type in Github such as a Fork or Push. The term, w represents the number of times user u performed Github action a on repo r at some timestep t such that $1 \leq t \leq T$. The term, T represents the latest timestep of G.

There are various ways Github temporal graph activity can be predicted. Most of the works covered in this review tackle the problem of predicting user-repo links spanning from time $T + 1$ up to $T + S$, given input features up to time T. S represents the number of desired future timesteps that one wishes to predict.

A couple of works have slight variations on this task. The *RA-DCRNN* [4] aims to only predict the number of activities a repo r has performed upon it from $T + 1$ to $T + S$. The user-repo interactions are not of concern. In the *DeepFork* paper [6], the authors aim to predict whether or not a link is created among a given *(user, repo, follower)* triplet, or (u, r, f_u), in which f_u is a user who follows u.

3.2. Measurement Granularities

An important point of consideration is how each prediction task is measured. This paper classifies prediction tasks and measurements into 2 main categories, *macroscopic* and *microscopic* levels. Firstly, there is macroscopic measurement. This involves measuring a prediction of a high-level phenomenon. For example, one could measure the accuracy of the predicted time series of tweets in Twitter, or the predicted time series of new user creation in Youtube. One could also measure some high-level aspect of the predicted network, such as the distance between the predicted and ground truth networks' degree distributions.

Secondly, there is the more fine-grained *microscopic* measurement, that involves measuring the low-level phenomenon of a network prediction, specifically "who does what when", or "who engages with whom when". In the former case, we are just concerned with what an individual does at time $T + S$. In the latter case, we are concerned with (1) who did what, (2) when, and (3) with whom (link prediction).

Note that in this work, all approaches covered involve predicting user or user-to-user activity at some future timestep, however, not all approaches utilize microscopic measurements. This is because for some datasets, it is not feasible to predict microscopic measurements with reasonable accuracy.

More details regarding how each approach measures accuracy will be given as they are described.

4. Challenges

There are various challenges involved with predicting activity on social media. One of the most obvious difficulties is that of the inherent issue of predicting the future. This issue can further be exacerbated by the issue of noisy historical data collection. If the historical data needed as initial conditions for temporal prediction is noisy or incomplete, it can be difficult or even impossible to predict future events with even some small degree of accuracy.

In [7], the authors explored the effect of data filtering on the ability to predict activity using two different cyber attack time series data sets. They showed that (1) the auto-correlation decreases at low sampling rates, (2) permutation entropy increases, and (3) the error of model-based techniques increase.

Another challenge in social media prediction is that of large data. There are can be potentially millions of users on various social media sites, so preprocessing steps or model training times can be greatly impacted by the overwhelming amount of data.

Lastly, too little data can be detrimental to model training as well. In [4], the authors noted the difficulty of predicting the time series of activity for infrequently active user-repo pairs. The authors note that the model simply predicted all 0's because the user-repo pair was not frequently active.

In this work, we will discuss how various previous social media prediction frameworks deal with these various challenges.

5. The Various Types of Github Prediction Frameworks

In this section we discuss the different approaches used for Github prediction. While reviewing the various literature, a hierarchy of framework approaches was observed. Figure 1 shows this hierarchy.

Firstly, we have the *Model Driven* vs. *Model Agnostic* framework categories. The *Model Agnostic* category contains 2 frameworks, as shown in Figure 2.

The *Model Driven* category contains more subcategories. It can be further divided into the *Decompositional* (Figure 3) and *Non-Decompositional* subcategories (Figure 4). Under the *Decompositional* category are the *Clustering-Based* and *Volume-to-User* subcategories (Figure 3). Under the *Non-Decompositional* category are the *Follower-Followee Driven*, and *Non-Follower-Followee Driven* approaches (Figure 4). The leaves of the hierarchy trees in Figures 2, 3, and 4 contain the different frameworks used in each category. In the following sections, these categories and frameworks will be discussed in more detail.

Note, we define framework as the simulation pipeline that is comprised of pipeline inputs, data pre-processing, model simulation, and pipeline output.

5.1. Model Driven vs. Model-Agnostic Frameworks

A *Model-Driven* approach is defined as prediction pipeline comprised of (1) data as input, and (2) a model that performs predictions using this input data. However, a *Model-Agnostic* approach is a prediction pipeline comprised of (1) data **and** a model as input, as well as (2) a program that performs predictions using both this model and data.

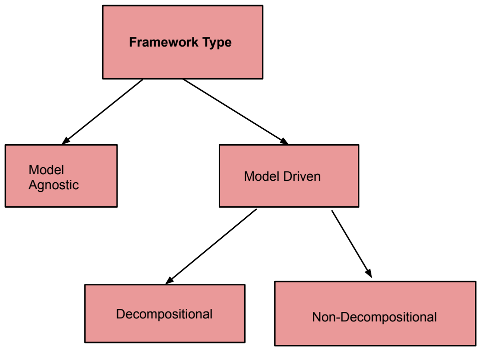

Figure 1: A diagram of the overall framework hierarchy discussed in this survey.

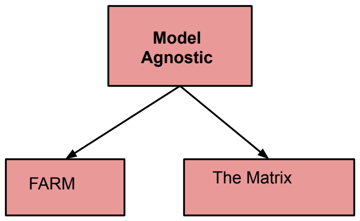

Figure 2: A diagram of the Model Agnostic approaches in this survey.

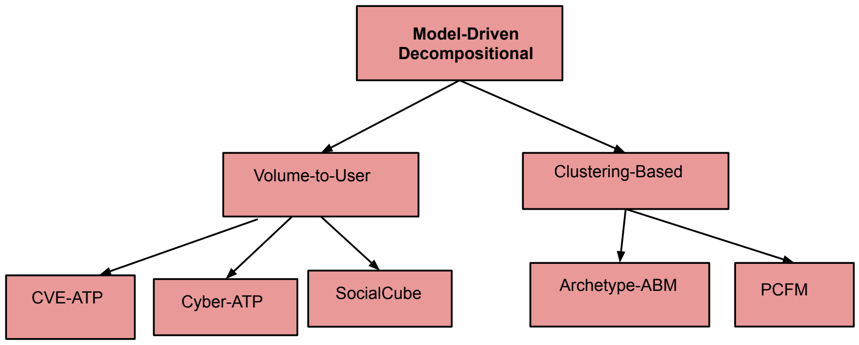

Figure 3: A diagram of the Model-Driven, Decompositional approaches in this survey.

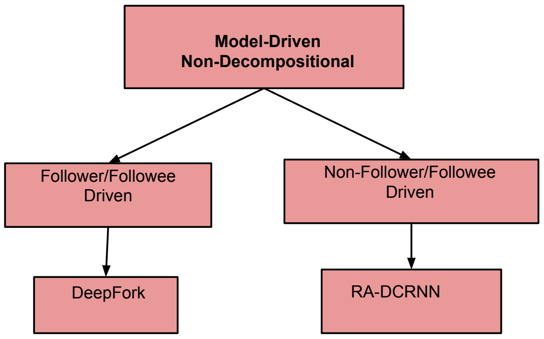

Figure 4: A diagram of the Model-Driven, Non-Decompositional approaches in this work.

In this survey, 7 of the 9 known Github prediction approaches are *Model-Driven*, and 2 of them are *Model-Agnostic*. These two are the *FARM* [8, 9, 10] and *Matrix* [11] frameworks. These 2 approaches are Distributed-Computing based. This means that they are programs made for a distributed computing system in which there is a controller process that communicates with multiple compute nodes. These Distributed-Computing frameworks emphasize usability and scalability.

The FARM [8, 9, 10] framework divides the simulation task among multiple computing nodes and uses a controller process in order to aggregate the final results and track progress. The Github graph is split up into multiple partitions using a graph partitioning algorithm and these partitions are placed into separate nodes for efficiency. The authors of [9] and [10] used 3 sampling-based agent models, a link prediction, and a Bayesian model with FARM and noted that by using FARM, they cut the runtime of these algorithms by 67% on average. It took FARM 20 minutes to simulate a Github network with 3 million users, 6 million repos, and 30 million events.

The authors of [11] introduced the *Matrix* Agent-Based Simulation Framework. Similar to FARM, this approach also utilizes a control process node connected to multiple computing nodes. In [11] a template is provided that a potential Github model must follow in order for it to be properly integrated.

The Github network is described as a discrete dynamical system $S(G, K, F, W)$. $G(V, E)$ is the Github graph with a set of nodes V and edges E. $K_t \in K$ is the array of *vertex* states at time t. Each state x_i maps to a node $v_i \in V$ The authors do not specify what is meant by *state*. Perhaps this is because Matrix is supposed to be "model agnostic". Therefore, *state* will be different depending on what you want to predict. For example, the state could be a feature vector describing a node v at time t, or in the case of a binary classification, the state could be a simple 1 or 0 to indicate whether or not a user was active at time t. F is an array of local functions for each node in V. Each function is used to predict the state of node v_i at time $t + 1$. Each node is mapped to its own function so that *Matrix* can perform parallel predictions for efficiency. Lastly, W is an update scheme function that sets the ordering for the node functions [11].

Matrix was able to simulate 3 million users, 13 million repos, and 239 million events in about 52 minutes [11].

5.2. Decompositional Approaches

Next, there are *Decompositional* prediction approaches. These include the approaches that break the main prediction task into smaller subtasks. There are 2 ways that the prediction tasks were decomposed in the literature. They are the *Clustering-Based* and *Volume-to-User* based approaches.

5.3. Clustering-Based Decompositional Approaches

In the *Decompositional Clustering-Based* approach, user-repo pairs are predicted by first clustering the users into different categories based on 1 or more attributes, and then using these clusters to make more informed predictions of which user-repo pairs were active at time $T + 1$.

For example, the Archetype-ABM [12] is a framework that aims to predict user-repository activity through clustering and user archetypes. In this approach, K Means clustering was used to divide users into 16 different groups based on their average monthly activity for 14 different Github events [12].

These clusters were known as "archetypes". These archetypes were then used to predict the actual users, repositories, and events at some future timestep $T + S$.

Another clusering-based approach is the Proposed Community Features Model (PCFM) [13], which is an agent-based approach that utilizes community clustering to predict user-to-repo activity. Each community is defined using a topic based approach using the profiles of the GitHub repositories in order to generate a fixed set of communities. Some examples of topics used include programming languages, operating systems, and profile keywords [13].

5.4. Volume-to-User Decompositional Approaches

The *Volume-to-User Decompositional* approaches in this work aim to predict user-repo interactions in a two step approach. First, the overall activity time series is predicted for a particular Github event. Then, these macroscopic activity counts, along with user-repo pair features are sent through a 2nd module to perform the more microscopic task of predicting user-to-repo activity over time. The 3 frameworks that use this approach are the *CVE-Action-to-Pair (CVE-ATP) Model* [3], the *Cyber-Action-to-Pair (Cyber-ATP) Model* [2] and the SocialCube model as described in [14] and [15].

The *CVE-ATP* and *Cyber-ATP* frameworks are 2 variations of the Action-to-Pair framework (ATP) [2, 3]. *ATP* is an LSTM neural network approach to Github user prediction. It decomposes the prediction problem into two tasks. Firstly, *ATP* predicts the daily-level volume of Github activities in the prediction time period of interest. This is the *Daily Level Prediction Task* [2]. It uses external features from Reddit and Twitter in order to make more accurate predictions. Then, *ATP*'s predicted daily counts, along with user-repo time series features are used to predict the number of events a user u performs on a repository r for each hour within each day of the prediction time frame of interest. This is the *Hourly User-Level Prediction Task* [2]. LSTM neural networks are used for both tasks.

The *ATP* framework was used to predict Github activity in two different datasets in two different works - [2] and [3]. With this in mind, we will refer to this framework as two different frameworks. The *ATP* implementation in [3] will be referred to as *CVE-ATP* because it was used to predict activity related to repositories in the Common Vulnerabilities and Exposures database (CVE). The data company Leidos used this database to create the CVE repo dataset.

The *ATP* implementation in [2] will be referred to as *Cyber-ATP* because it was used to predict activity related to cybersecurity repositories. This data was also gathered by Leidos. They mined the text of issue comments in Github and added a repo to the cybersecurity list if the repo's associated issue comments contained keywords related to cybersecurity such as "security" or "bot", etc.

The Socialcube [14, 15] model predicts user-to-repo Github activity in two steps. First, an ARIMA model is used to predict the overall activity time series for 10 different Github events. Then, an ARIMA model is used to predict the activity time series of each user-repo pair.

5.5. Non-Decompositional Approaches

The *Non-Decompositional* approaches do not break the overall prediction task into subtasks. Instead, they directly predict the node or edge pair activity at time $T + 1$. There are 2 types, *Follower-Followee Driven* and *Non-Follower-Followee Driven* approaches.

5.6. Follower-Followee Driven, Non-Decompositional Approaches

In the *Follower-Followee* driven approach, the framework predicts user and repo activity at time $T + 1$ by explicitly leveraging the relationships between a user, u, a follower of user u, f_u, and a repo, r.

The approach in this category is the DeepFork Framework [6]. It is a neural network based approach that treats the user-repo prediction task as a binary classification. DeepFork attempts to predict "information diffusion" among a user, follower, and repo triplet at time $T + 1$. In other words, instead of predicting whether (u, r) forms a link at $T + 1$, it predicts whether a link is formed among (u, r, f_u) at $T+1$. In this case, f_u is a follower of u. This link represents the act of user u performing an action on repo, r; follower, f_u seeing this action, and then f_u also performing an action on repo r. In Github, it is possible for f_u to know u's actions because users can follow one another.

5.7. Non-Follower-Followee-Driven, Non-Decompositional Approaches

The term *Non-Follower-Followee-Driven Approach*, refers to any type of framework that does not explicitly model the dynamics between (u, r, f_u) triplets. Instead, a framework in this category directly predicts the node or edge activity at time $T + 1$ using some other means.

The work in the *Non-Follower-Followee* category is the *Repo Activity Diffusion Convolution Recurrent Neural Network* model, or *RA-DCRNN* of [4]. This model was used to perform temporal link predictions on a Github CVE (Common Vulnerabilities and Exposures) dataset. The dataset contains information pertaining to CVE exploits posted in cyber-security-related Github repositories.

In this work, the authors modelled the Github network as a homogenous, undirected network, in which all nodes were repos. Note this is different from the other works mentioned in which nodes were modelled as both users and repos. In this particular dataset, an edge exists between two CVEs in a given timestep t if both repos contain Github events pertaining to the same CVE identifier [4].

The *RA-DCRNN* is a Diffusion Convolutional Recurrent Neural Network. Each timestep represents the daily granularity, and the prediction task was as follows: Given the 7-day history time series for a repo, r, predict the number of Push Events performed on this repo over the next 7 days.

The Diffusion Convolutional Recurrent Neural Network is a recurrent neural network that utilizes two popular neural network layers, Gated Recurrent Units (GRU) [16] and Graph Convolutional Networks (GCNs) [17]. These types of networks are useful for performing sequence to sequence predictions with graph data.

5.8. Github Data Summary

Table 1 contains prediction task and test set count-related information for each of the different Github prediction frameworks. The *What is Predicted* column shows the prediction task. The *Users*, *Repos*, and *Events* columns show the number of those elements in the test set.

Table 2 contains time-related information for the test sets used for each Github prediction framework. The *#Timesteps in Testing Period* column shows the number of timesteps in each testing period. The *Timestep Granularity* column shows the granularity of each time step. The *Prediction Period Dates* column shows the timespan of prediction. Lastly, the *Prediction Runtime* column shows how long it took the framework to perform the prediction. For example, the *FARM* framework was tested on the period spanning from 2/1/2018 to 2/28/2018. There were 28 days in this period and predictions were at daily granularity. Lastly, note that any value that a given paper did not report was marked as "N.M.", which stands for "Not Mentioned".

6. Model Performance Results

There are many ways to evaluate success within the domain of Github activity prediction. In this section, the various ways of doing so are discussed.

6.1. Model-Agnostic Results

The *FARM* framework was used to simulate the performance of 5 different models - a Bayesian model, a link prediction model, and 3 sampling-based models [9]. These models were evaluated on how well they predicted the popularity of the users and repositories. Rank Biased Overlap *(RBO)*, R^2, and *RMSE* metrics were used. The authors of [9] noted the sampling models performed the best. For example, for the R^2 metric for measuring the event count for issues per repository, the 3 sampling models all had R^2 scores of 0.74, 0.74, and 0.75; while the Bayesian and link prediction models had scores of 0.05 and 0.58, respectively (higher is better).

Table 1: Table of test set prediction task and count-related information for the Github prediction frameworks. *N.M.* stands for *Not Mentioned*. It means that the authors of the paper did not state the value in the given cell.

Github Test Set Count Information				
Framework	**What is Predicted**	**#Users**	**#Repos**	**#Events**
FARM [8, 9, 10]	# user-repo activities	3 mil	6 mil	30 mil
Matrix [11]	# user-repo activities	3 mil	13 mil	239 mil
Archetype-ABM [12]	# user-repo activities	N.M.	N.M.	N.M.
PCFM [13]	# user-repo activities	N.M.	N.M.	N.M.
CVE-ATP [3]	# user-repo activities	N.M.	N.M.	N.M.
Cyber-ATP [2]	# user-repo activities	2 mil	400,000	65 mil
SocialCube [15, 14]	# user-repo activities	N.M.	1000	1.4 mil
DeepFork [6]	user-repo-follower classification	N.M.	N.M.	N.M.
RA-DCRNN [4]	# of repo activities	n/a	22,052	36,584

Table 2: Table of test set time-related information for the Github prediction frameworks. *N.M.* stands for *Not Mentioned*. It means that the authors of the paper did not state the value in the given cell.

Github Test Set Time and Date Information				
Framework	**#Time-steps in Testing Period**	**Timestep Granularity**	**Prediction (Test) Period Dates**	**Prediction Runtime**
FARM [8, 9, 10]	28	daily	2/1/2018 to 2/28/2018	20 min
Matrix [11]	14	daily	N.M.	52 min
Archetype-ABM [12]	N.M.	N.M.	N.M.	N.M.
PCFM [13]	N.M.	weekly	N.M.	N.M.
CVE-ATP [3]	744	hourly	8/1/2017 to 8/31/2017	N.M.
Cyber-ATP [2]	744	hourly	8/1/2017 to 8/31/2017	3 hours
SocialCube [15, 14]	31	daily	7/1/2015 to 7/31/2015	6 hours
DeepFork [6]	1	monthly	1/1/2017 to 2/1/2017	N.M.
RA-DCRNN [4]	237	daily	8/7/2017 to 3/31/2018	N.M.

The *Matrix* framework was not evaluated on accuracy metrics, but was instead evaluated on runtime performance under multiple conditions. They found that runtime decreased approximately linearly with the addition of more CPU cores. Furthermore, they found that runtime performance increased approximately linearly with the addition of more users. The smallest number of users *Matrix* simulated was 300,000, and that took about 6.6 minutes. The largest number of users simulated was 3 million, which took about 52 minutes.

6.2. Volume-to-User Results

For the *CVE-ATP* model, the authors measured the distribution of Fork and Watch events across Github repositories [3]. They noted that both distributions of events were close to the ground truth distributions. The JS divergence scores for Forks and Watches were 0.0029 and 0.0020, respectively (lower is better). For the R^2 metric, the scores were 0.6300 and 0.6067, respectively [3].

In the *Cyber-ATP* model in [2], the authors used a myriad of measurements and metrics to measure how well the overall simulated pattern of user-to-repo Github activity matched that of the ground truth. The model performed particularly well at predicting *Repo activity disparity* for Fork events. This is a measure of how well the simulated and ground truth Fork event distributions match up, and it was measured with Absolute Difference. The *Cyber-ATP* model's absolute difference was around 1, while the baseline "shifted" model's absolute difference was much higher, at 4.

SocialCube predicted the Github PushEvent, PullRequest, and ForkEvent 31-day time series using ARIMA [14]. *SocialCube* then used these initial predictions to perform the more fine-grained task of predicting the "affinity rates" of all the user-repo pairs. The affinity rate is the rate (between 0 and 1) at which a user u interacts with a repo r within the overall 31-day prediction period. The authors of [14] used Absolute Error to measure the ground truth affinity rates with the predicted affinity rates. They found that *SocialCube* strongly outperformed the baseline "Stationary" model, which was constructed by shifting the past 31 days of affinity rates into the next 31 days. The Socialcube model had a smaller absolute error than the baseline model for 78.6% of the users. The baseline model had smaller absolute error for only 21.4% of users.

6.3. Clustering-Based Results

The *Archetype-ABM* model was used to predict which repo a user would engage with. The authors of [12] used a parameter, σ to control the number of candidate repos that *Archetype-ABM* could choose from when predicting a repo for a particular user. In their experiments, the authors tested with various values of σ ranging from 1 to 32 [12].

The performance of *Archetype-ABM* was evaluated by measuring, with the Gini Coefficient, how well (1) activities were distributed among users and (2) how well activities were distributed among repos. They divided the activties into 3 categories: (1) Contributions, (2) Watches, and (3) Forks. Contributions are comprised of all Github activities that are not Forks or Watches such as Pushes, Pulls, etc. As a baseline model, the authors of [12] used a *mean model*, which predicts the number of activities in the future by using the historical mean of activities. Overall, the *Archetype-ABM* outperformed the baseline model. *Archetype-ABM* performed particularly well at predicting the distribution of Contribution Events over users. When $\sigma = 32$, *Error of Gini coefficient* for the baseline was around 0.5, while the error for *Archetype-ABM* was less than 0.1.

Archetype-ABM also outperformed the baseline when predicting the distribution of events over repos, albeit by not as much. When $\sigma = 32$, the baseline had an error of around 0.125 while *Archetype-ABM* had an error of around 0.1.

The *Proposed Community Features Model (PCFM)* [13] measured performance on a wide variety of metrics that fell into 4 different resolutions: (1) user (2) content (repo), (3) community (a subset of users), and (4) population (the entire Github network). The metrics used were (1) RMSE, (2) KS-test, (3) JS-Divergence, (4) Absolute Difference, and (5) Rank-Biased Overlap (RBO). *PCFM* was then compared with 3 other baseline models. One overall metric score was calculated for each model by normalizing all metric results between 0 and 1 by measurement group and metric type. This final score was called the "normalized metric error" [13]. *PCFM* was found to have the lowest error scores out of the 4 models. For example, for the "user" category *PCFM* model had a normalized metric error of around 0.2, while the best baseline had an error of around 0.4.

6.4. Follower/Followee Driven Results

The one Github model that performed a classification task was the *DeepFork* model [6]. Recall the prediction task was to predict whether a followee-follower-repo triplet would be active at future timestep $T+1$. *DeepFork* outperformed its baselines, with an average F1 score around 70%. Perhaps the reason it achieved this somewhat high score is due to the setup of the prediction task. The authors used time T as the training period and it ranged from August 1 2016 to Dec 31 2016 (5 months). The granularity of $T+1$ (the test period) was 1 entire month (Jan 1 2017 to Feb 1 2017). So, the task was for DeepFork to predict if a triplet would be active anytime within an entire month. Perhaps if $T+1$ was a smaller granualrity (such as 1 day), the accuracy may have been much lower.

6.5. Non-Follower/Followee Driven Results

Recall that for the *RA-DCRNN* model [4], the authors predicted sequences of repo-to-repo adjacency matrices over 7-day periods. They used RMSE and MAE of each graph instance to measure success. In order to alleviate the data sparsity issue of repo time series, they clustered repos according to different characteristics. However, they found that with fewer clusters of repo nodes, the metric performance became worse. For example, the repo partition with the fewest nodes had an MAE and RMSE of 6.37 and 29.92, respectively. However, the partition with the largest number of clusters of nodes, had an MAE and RMSE of 1.41 and 7.07, respectively. However, the authors do note that the model trained on the smaller number of clusters dataset was best at predicting spikes in the time series. The problem was that these predicted spikes did not perfectly line up with the ground truth time series. So, perhaps different metrics could have been used to account for this fact [4].

7. Real World Applications

In this section we discuss how each of the prediction frameworks could be applied to real-world scenarios.

7.1. Github Framework Real World Applications

There are various practical applications for the Github frameworks depending on the goal of the end user. As mentioned earlier, perhaps the user may be interested in what technologies would become popular in the near future. To that end, it may be useful to use a framework that predicts the popularity of a repo over time. The *RA-DCRNN* [4] obviously meets that goal, because it directly predicts the activities performed upon various repos over time. Most of the approaches that predict user-to-repo pairs would be useful as well, one would of course just simply ignore the final user predictions made by these models and simply pay attention to the repo predictions. The *Archetype-ABM* [18] may be less useful in this case because the authors noted that while it does predict user-repo pairs, it performs much better at predicting users and much worse at predicting repos. Furthermore, the *DeepFork* [6] model may be of less interest as well, because it performs a binary classification (0 or 1) with "0" meaning a repo will be acted upon by a user and 1 otherwise. So, it does not contain any information about the popularity of a repo. Therefore, the models that would be of interest for repo popularity prediction would be the *CVE-ATP* [3], *Cyber-ATP* [2], *SocialCube* [15], *PCFM* [13], and *RA-DCRNN* [4] models.

The other use of Github prediction frameworks would of course be to detect a cyber attack or exploit of a software vulnerability before it occurs. To that end, it would be useful to predict either future (1) user activity, (2) repo activity, or (3) user-repo activity. In that case, all the Model-Driven approaches would be appropriate. Models that predict all 3 may be preferable because one could monitor 2 scenarios: (1) tracking a user or group of users whose activity is usually correlated with malicious events or (2) tracking the increased amount of activity on repos known to contain software vulnerabilities or associated with cyber attacks. The *CVE-ATP*, *Cyber-ATP*, *SocialCube*,*Archetype-ABM* and *PCFM* models would be useful for obtaining these 3 pieces of information because these models predict user-repo pairs. The *RA-DCRNN* model could be useful, but it only captures repo popularity.

DeepFork [6] predicts user-follower-repo triplets as a binary classification. It may not be useful if the volume of activities is desired, however, something that it does that the other models do not is track not only the user who performs an activity on a repo, but it also predicts if a follower of that user will also perform an acitivty upon a repo, which could be a powerful tool if one wanted to predict the influence of different users upon each other.

With some potential adjustments, the *Model-Agnostic* approaches, FARM [9] and the Matrix [11] could be used in conjunction with any of the *Model-Driven* approaches if scalability becomes an issue.

8. Future Work

The direction of future work depends greatly on the framework of interest. In this section, we discuss the different possibilities of future work in detail. Generally, all approaches have room to improve their accuracy.

8.1. Model Agnostic Future Work

In [8], the creators of the FARM Architecture note that further runtime improvements could be made to FARM if clusters of users that share a similar or identical attributes could be found. If such users could be identified, then the representation of these users could be compressed so that the simulations are more memory efficient.

The authors of the Matrix [11] note that an outstanding issue with the approach is evaluation of the trade-off between model fidelity and accuracy of models. Future work with the Matrix model would involve more ways of understanding this tradeoff and perhaps even mitigating it.

8.2. Model-Driven - Volume to-User Future Work

A limitation of the ATP models in [2] and [3] is that they can only predict activity of user-to-repo pairs that have occurred in the past. Future work for these models would involve resolving this issue.

The SocialCube [15] model does predict new users, so instead, for future work explorations, the authors discuss performance optimization as well as exploring the use of other user-prediction models.

8.3. Model-Driven - Clustering Based Future Work

The authors of the PCFM model [13] note that the main weakness of their approach is that it does not provide an explanation for the users' behavior nor does it capture peer to peer influence. Perhaps future improvements to this model could involve analysis into the types of users being predicted, and a mechanism that captures peer to peer influence.

In [18], the creators of the Archetype-ABM model note that their model scores well on the user-prediction tasks, but less well on the repo-prediction tasks. This is perhaps due to the fact that the Archetype-ABM creates archetypes based on users and not on repos. So, for future work, the authors propose an approach that also creates repo-archetypes as well to improve repo-prediction.

8.4. Model-Driven - Non-Decompositional Future Work

The creators of *DeepFork* [6] found much success in using hand-crafted topological and node features to predict user-follower-repo triplets. Future work for this type of model could involve further analysis of why certain topological and node features worked for the model, and if they give any insights into the dynamics of user-follower-repo triplets.

The authors of the *RA-DCRNN* [4] model was able to outperform other baseline models in the task of predicting repo activities, but the authors noted that the model still struggled to predict future activity for lowly-active users. Future work in this area could involve creating a DCRNN architecture that specializes in sparse activity prediction.

9. Conclusion

In this review we discussed recent Artificial Intelligence-driven Github user-level prediction approaches. We discussed how within the realm of Github, the main approaches can be divided into the *Model Agnostic* and *Model-Driven* categories. Within the *Model-Driven* approaches, the different types discussed were *Volume-to-User*, *Clustering-Based*, *Follower/Followee Driven*, and *Non-Follower/ Followee Driven.*

We also discussed the challenges of social media prediction, such as scalability, noisy or uncertain data, and sparse data. Furthermore, we discussed how these different approaches performed, possible real-world applications, and potential directions for future work.

This review can be used to guide researchers who are interested in creating AI-driven user-level prediction frameworks. Although we focused on Github prediction, one should note that the ideas presented in these works are applicable (with perhaps some modifications) to other temporal network prediction domains, such as those found in more typical kinds of social media networks.

10. Acknowledgements

This work was partially supported by the DARPA and Air Force Research Laboratory via contract FA8650-18-C-7825.

References

[1] Github homepage.
URL `https://github.com/`

[2] R. Liu, F. Mubang, L. O. Hall, S. Horawalavithana, A. Iamnitchi, J. Skvoretz, Predicting longitudinal user activity at fine time granularity in online collaborative platforms, in: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019, pp. 2535–2542. `doi:10.1109/SMC.2019.8914586`.

[3] S. Horawalavithana, A. Bhattacharjee, R. Liu, N. Choudhury, L. O. Hall, A. Iamnitchi, Mentions of Security Vulnerabilities on Reddit, Twitter and GitHub, in: Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'19), Thessaloniki, Greece, 2019.

[4] A. Hernandez, K. W. NG, A. Iamnitchi, Using Deep Learning for Temporal Forecasting of User Activity on Social Media: Challenges and Limitations, in: Proceedings of Temporal Web Analytics Workshop, Companion Proceedings of The 2020 World Wide Web Conference (TempWeb'20), Taipei, Taipei, 2020.

[5] T. Brewster, How hackers broke equifax: Exploiting a patchable vulnerability, Forbes.
URL `https://www.forbes.com/sites/thomasbrewster/\2017/09/14/equifax-hack-the-result-of-\patched-vulnerability/?sh=b6a317d5cda4`

[6] R. Akula, I. Garibay, N. Yousefi, Deepfork: Supervised prediction of information diffusion in github, in: Conference: Proceedings of the International Conference on Industrial Engineering and Operations ManagementAt: Bangkok, Thailand, March 5-7, 2019, 2019.

[7] N. Tavabi, A. Abeliuk, N. Mokhberian, J. Abramson, K. Lerman, Challenges in forecasting malicious events from incomplete data, in: Companion Proceedings of the Web Conference 2020, WWW '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 603–610. `doi:10.1145/3366424.3385774`.
URL `https://doi.org/10.1145/3366424.3385774`

[8] J. Blythe, A. Tregubov, Farm: Architecture for distributed agent-based social simulations, in: D. Lin, T. Ishida, F. Zambonelli, I. Noda (Eds.), Massively Multi-Agent Systems II, Springer International Publishing, Cham, 2019, pp. 96–107.

[9] J. Blythe, J. Bollenbacher, D. Huang, P.-M. Hui, R. Krohn, D. Pacheco, G. Murić, A. Sapienza, A. Tregubov, Y.-Y. Ahn, A. Flammini, K. Lerman, F. Menczer, T. Weninger, E. Ferrara, Massive multi-agent data-driven simulations of the github ecosystem, in: PAAMS, 2019.

[10] J. Blythe, E. Ferrara, D. Huang, K. Lerman, G. Murić, A. Sapienza, A. Tregubov, D. Pacheco, J. Bollenbacher, A. Flammini, P.-M. Hui, F. Menczer, The darpa socialsim challenge: Massive multi-agent simulations of the github ecosystem, in: AAMAS, 2019.

[11] P. Bhattacharya, S. Ekanayake, C. Kuhlman, C. Lebiere, D. Morrison, S. Swarup, M. Wilson, M. Orr, The matrix: An agent-based modeling framework for data intensive simulations, in: AAMAS, 2019.

[12] S. Saadat, C. Gunaratne, N. Baral, G. Sukthankar, I. Garibay, Initializing agent-based models with clustering archetypes, in: R. Thomson, C. Dancy, A. Hyder, H. Bisgin (Eds.), Social, Cultural, and Behavioral Modeling, Springer International Publishing, Cham, 2018, pp. 233–239.

[13] N. Hajiakhoond Bidoki, M. Schiappa, G. Sukthankar, I. Garibay, Modeling social coding dynamics with sampled historical data, Online Social Networks and Media 16 (2020) 100070. `doi:10.1016/j.osnem.2020.100070`.

[14] T. Abdelzaher, J. Han, Y. Hao, A. Jing, D. Liu, S. Liu, H. Nguyen, D. Nicol, H. Shao, T. Wang, S. Yao, Y. Zhang, O. Malik, S. Dipple, J. Flamino, F. Buchanan, S. Cohen, G. Korniss, B. Szymanski, Multiscale online media simulation with socialcube, Computational and Mathematical Organization Theory 26 (06 2020). `doi:10.1007/s10588-019-09303-7`.

[15] S. Yao, Y. Hao, D. Liu, S. Liu, H. Shao, J. Wu, M. Bamba, T. Abdelzaher, J. Flamino, B. Szymanski, A predictive self-configuring simulator for online media, in: 2018 Winter Simulation Conference (WSC), 2018, pp. 1262–1273. `doi:10.1109/WSC.2018.8632412`.

[16] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, in: NIPS 2014 Workshop on Deep Learning, December 2014, 2014.

[17] T. N. Kipf, M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, in: Proceedings of the 5th International Conference on Learning Representations, ICLR '17, 2017.
URL `https://openreview.net/forum?id=SJU4ayYgl`

[18] S. Saadat, C. Gunaratne, N. Baral, G. Sukthankar, I. Garibay, Initializing Agent-Based Models with Clustering Archetypes, 2018, pp. 233–239. `doi:10.1007/978-3-319-93372-6_27`.