The DARPA SocialSim Challenge: Massive Multi-Agent Simulations of the Github Ecosystem

Extended Abstract

James Blythe, Emilio Ferrara, Di Huang, Kristina Lerman, Goran Muric, Anna Sapienza, Alexey Tregubov, Diogo Pacheco, John Bollenbacher, Alessandro Flammini, Pik-Mai Hui, Filippo Menczer
{blythe,ferrarae,dihuang,lerman,gmuric,annas,tregubov}@isi.edu; {pacheco,jmbollen,aflammin,huip,fil}@iu.edu
USC Information Sciences Institute, Marina del Rey, CA (USA); Indiana University, Bloomington, IN (USA)

ABSTRACT

We model the evolution of GitHub, a large collaborative software-development ecosystem, using massive multi-agent simulations as a part of DARPA's SocialSim program. Our best performing models and our agent-based simulation framework are described here. Six different agent models were tested based on a variety of machine learning and statistical methods. The most successful models are based on sampling from a stationary probability distribution of actions and repositories for each agent.

ACM Reference Format: J. Blythe, E. Ferrara, D. Huang, K. Lerman, G. Muric, A. Sapienza, A. Tregubov, D. Pacheco, J. Bollenbacher, A. Flammini, P.-M. Hui, F. Menczer. 2019. The DARPA SocialSim Challenge: Massive Multi-Agent Simulations of the Github Ecosystem. In *Proc. of the 18th Int'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS'19), Montreal, Canada, May 13–17, 2019, IFAAMAS, 3 pages.*

1 INTRODUCTION

The DARPA SocialSim challenge problem measured participant's ability, given 30 months of meta-data on user activity on GitHub, to predict the next months' activity as measured by a broad range of metrics applied to ground truth, using agent-based simulation. The challenge involved making predictions about roughly 3 million individuals performing a combined 30 million actions on 6 million repositories. We describe the agent framework and the models we employed. Our team used a variety of learning methods contributing to six different types of agents that were tested against a wide range of metrics. The broadly most successful method of those tried sampled from a stationary probability distribution of actions and target repositories for each agent.

First, we describe the agent-based simulator we developed to carry out massive-scale simulations of techno-social systems. Second, we present the inference methods that we employed to implement different agent-based models, based on statistical modeling of historical activity, graph embedding to infer future interactions, Bayesian models to capture activity processes, and methods to predict the emergence of new users and repositories that did not exist in the historical data. These are novel applications of existing analytical tools to derive agent models from available data. Third, we provide a rigorous evaluation of the performance of six different models, as measured by a wide range of metrics. We also describe the DARPA SocialSim GitHub Challenge, provide a characterization of its rules, and describe how our team tackled it.

Our platform and models are general in scope, and have also been applied to large-scale agent simulations of behavior on the Twitter and Reddit social media platforms.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2 CHALLENGE PROBLEM DESCRIPTION

The DARPA SocialSim Challenge aims to simulate specific types of interactions between users and repositories on GitHub. In detail, it focuses on the simulation of social structure and temporal dynamics of the system, as well as looking at individual, community and population behaviors. We model ten event types a user can perform on a given repository: *Create*, *Delete*, *Push*, *Pull*, *Issue*, *Issue-Comment*, *Pull-Request*, *Commit-Comment*, *Watch*, and *Fork*.

The training set are all events of public users and repositories in the period spanning from 8/1/17 to 8/31/17, and 1/17/18 to 1/31/18, as well as metadata such as repository languages, user types etc. This includes a total of about $2.0M$ users and $3.3M$ repositories. For the challenge, we simulate the events, users, and repositories of GitHub from 2/1/18 to 2/28/18. As the training set included a gap of 4.5 months, additional information about the state of the system was provided, including all the profiles from users and repositories that were created during the gap.

3 AGENT FRAMEWORK AND DOMAIN IMPLEMENTATION

To implement our agent models we used FARM—an agent-based simulation framework implemented in Python that supports large-scale distributed simulations [5]. FARM also keeps track of the repeated and systematic experimentation required to validate the results from multi-agent simulations. FARM supports agents developed with the DASH framework [3], although it may be used with any agent through an API.

In our experiments, DASH agents represent GitHub users and implement GitHub events. Agents in FARM can communicate either directly or by taking actions that are sent to a shared state object, called a hub, that can be observed by other agents. In the GitHub simulation model, every action taken by a user acts on a repository, so communication is modeled indirectly by sending actions to a hub that maintains the state of a set of repositories and provides information to agents about their repositories of interest.

To support scaling to millions of agents and repositories FARM provides a multi-process infrastructure to divide agents and state across multiple hosts [5]. One hub is present on each image and

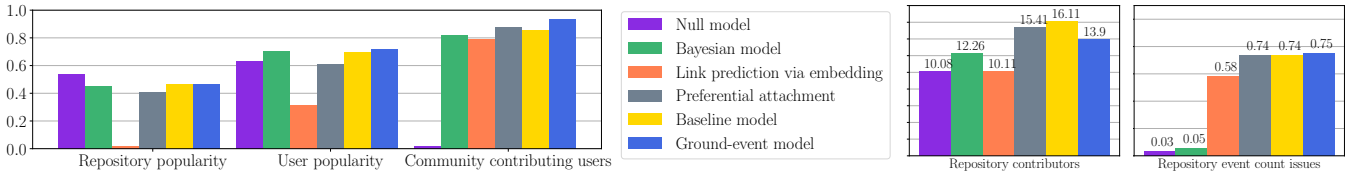

Figure 1: Left: Popularity metrics, *RBO*, and community contributing users, higher is better. Center: Repository contributors, *RMSE*, lower is better. Right: event issue count, R^2, higher is better.

shared state is managed with Apache ZooKeeper [1]. Using a multi-level graph partitioning algorithm to minimize the amount of communication across partitions, simulation time was reduced by 67% [5].

4 AGENT MODELS

Stationary probabilistic models. In the stationary probabilistic models, each agent's actions are determined by a stationary probability distribution built from the past history of events the agent has initiated. The overall event rate and the probability of each action are computed individually for each agent. We implemented three variations of probabilistic simulation models: the *baseline model* selects an event type and independently selects the repository on which the selected action is to be applied; the *ground-event model* selects an event type and repository simultaneously; the *preferential attachment model* extends the baseline model by redefining agent behaviour for *watch* and *fork* events. In all models, the frequency of agents' actions is determined by the event rate observed in the past for each user.

Link prediction through embedding. Here we formulate the problem of predicting user-repository interaction as a link prediction task, by describing our system as a bipartite network in which each node is either a user or a repository and links in each network are specific events. We generate a bipartite network for each event type with the exception of *Create* and *Delete* events. We then represent each of the built networks as a weighted adjacency matrix $\mathbf{A}_e \in \mathbb{R}^{|U|\times|R|}$, where e is an event type. Given the matrix $\mathbf{A}_e$ for each event type e, we compare embedding methods [8] against a random baseline: Graph Factorization (GF), Laplacian Eigenmaps (LE), and Hybrid Orthogonal Projection and Estimation (HOPE). We test performance using the MeanAveragePrecision (MAP), which estimates a model precision for each node and computes the average over all nodes. All the methods outperform our random baseline, which predicts links in a random fashion. In the experiments below we used GF, which combined good performance with scalability.

Bayesian model. The GitHub Challenge can be seen as finding relationships between the three governing entities: users, repositories, and events. We empirically measured the probabilities of these relationships to adjust the posterior probabilities of a generative model. In general terms, the model first chooses whether to create a new user or to select an existing one. Then, it decides between a category of events. Finally, it selects a repository and an action to perform. We investigated the trade-off between recency and history as driving forces to popularity [2]. The results showed that *less is more* in terms of the amount of data needed to predict users' activity level. The user selection implements a rank model [7] based on user's activity level, with past activity being less weighted using a 30-day half-life decay.

Modeling new users and repositories. We build a parsimonious model, able to predict the frequency of a particular event type e performed by user u on a repository r, conditioned by non-existing history of interaction between u and r. We compiled a total list of 124 features extracted for a sample of user-repository pairs and then we employ the Structured Sum of Squares Decomposition (S3D) algorithm [6] to rank the features by their importance. The ranks of features differ among the models. Still, the most informative features for predicting the actions of a user to a repository, are derived from their mutual relation. However, many actions depend on the information of the repository ownership.

5 CANDIDATE AGENTS AND RESULTS

We developed GitHub user agents that implemented the following models described in the previous section: (1) the null model, (2) the probabilistic baseline model, (3) the probabilistic ground-event model, (4) the preferential attachment model, (5) link prediction via embedding (LPE), and (6) the Bayesian model. The null model is just a shift of the past data to the future.

To answer various research questions of the DARPA SocialSim Challenge more than a dozen metrics were used to evaluate our simulation results. Figure 1 on the left shows evaluation results for two bounded metrics: repository popularity, user popularity. All metrics are scaled to the [0,1] interval, higher is better. In the center, Figure 1 shows repository contributors - the number of daily unique contributors to a repository as a function of time. It is calculated as *RMSE*, lower is better. On the right, it also shows repository event count issues - the number of issue events by repository. It is calculated as R^2, higher is better. Other metrics are discussed in [4].

We demonstrated novel agents built using six different learning principles to predict the future behavior of GitHub based on training data. Across a broad array of prediction metrics, no single approach dominated the others. One interesting constant was that, since overall behavior is constantly changing, it was detrimental to use all available training data in building the agents. Instead, one month of data proved optimal across most of the agents, although this precise number is no doubt dependent on the social network.

The main contribution of this work is to develop a framework for massive-scale simulations in which agents embodying very different ideas about decision making and data use can be directly compared. Our approach is general, and has recently been applied to the Reddit and Twitter social networks. Among other directions, we are considering ways to combine these agent models, both intra-agent, combining some of the best features of different approaches in a single agent, and inter-agent, with simulations of more than one type of agent, based on their predicted role.

Acknowledgements. Work supported by DARPA (W911NF-17-C-0094).

REFERENCES

[1] Apache. 2018. Apache ZooKeeper. https://zookeeper.apache.org. (2018).

[2] Hugo Barbosa, Fernando B. de Lima-Neto, Alexandre Evsukoff, and Ronaldo Menezes. 2015. The effect of recency to human mobility. *EPJ Data Science* 4, 1 (2015), 1–14. arXiv:1504.01442

[3] James Blythe. 2012. A dual-process cognitive model for testing resilient control systems. In *5th International Symposium on Resilient Control Systems*. 8–12.

[4] James Blythe, Emilio Ferrara, Di Huang, Kristina Lerman, Goran Muric, Anna Sapienza, Alexey Tregubov, Diogo Pacheco, John Bollenbacher, Alessandro Flammini, Pik-Mai Hui, and Filippo Menczer. 2019. Massive Multi-Agent Data-Driven Simulationsof the GitHub Ecosystem. In *International Conference on Autonomous Agents and Multiagent Systems PAAMS*.

[5] James Blythe and Alexey Tregubov. 2018. FARM: Architecture for DistributedAgent-based Social Simulations. In *IJCAI/AAMAS Workshop on Massively Multi-Agent Systems*.

[6] Peter G Fennell, Zhiya Zuo, and Kristina Lerman. 2018. Predicting and Explaining Behavioral Data with Structured Feature Space Decomposition. (2018). arXiv:1810.09841

[7] Santo Fortunato, Alessandro Flammini, and Filippo Menczer. 2006. Scale-free network growth by ranking. *Physical Review Letters* 96, 21 (2006), 218701.

[8] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.