Predicting Popularity of Open Source Projects Using Recurrent Neural Networks

Sefa Eren Sahin, Kubilay Karpat, Ayse Tosun

Predicting Popularity of Open Source Projects Using Recurrent Neural Networks

Sefa Eren Sahin, Kubilay Karpat, and Ayse Tosun

Faculty of Computer and Informatics Engineering, Istanbul Technical University, 34469, Turkey {sahinsef, karpatk, tosunay}@itu.edu.tr

Abstract. GitHub is the largest open source source development platform with millions of repositories on variety of topics. The number of stars received by a repository is often considered as a measure of its popularity. Predicting the number of stars of a repository has been associated with the number of forks, commits, followers, documentation size, and programming language in the literature. We extend prior studies in terms of input features and algorithm: We define six features from GitHub events corresponding to the development activities, and additional six features incorporating the influence of users (followers and contributors) on the popularity of projects into their development activities. We propose a time-series based forecast model using Recurrent Neural Networks to predict the number of stars received in consecutive k days. We assess the performance of our proposed model with varying k (1,7,14,30 days) and with varying input features. Our analysis on five topmost starred repositories in data visualization area shows that the error rate ranges between 19.76 and 70.57 among the projects. The best performing models use either features from development activities only, or all metrics including all the features.

Keywords: open source projects · predicting stars · recurrent neural networks.

1 Introduction

GitHub is a web-based collaborative, software development platform built on the *git* version control system; it is also considered as a social community of developers [13]. In GitHub, users can follow the projects that they are interested in through GitHub stargazers [14]. GitHub keeps millions of repositories on variety of topics. The number of projects are generally high on popular topics such as machine learning, visualization libraries, web application platforms. Since there are many similar projects specialized on the same topic that a user can follow, it is necessary to make a decent analysis on the existing alternatives in order to find the most suitable library or tool for a specific requirement of a user. The number of GitHub stars can be an appropriate criterion for this selection, since it is viewed as a measure of others developers' approval on the suitability of a project for a specific task, or in other words, a project's popularity [17].

Long-term support is an important factor while evaluating both OSS and commercial software projects, however OSS projects do not offer any binding contract about any future updates and maintenance. At this point, forecasting the number of stars of those OSS projects could be helpful. For example, Borges et al. [3] built a forecast model for GitHub projects based on the number of forks, commits and contributors, code features (age, programming language) and documentation of the projects. Incorporating other features from issues and users [6], [1] into the development activity of those projects produced more valuable insights in other studies. Models on predicting the number of stars are often built with time-series data using different regression techniques [3], [17]. We extend those models by adding both events taken from GitHub API such as number of forks, releases, commits, and issues, and weighting them based on the users' (developers and followers) influence. We also use a special variation of Recurrent Neural Networks (RNNs) which has been popularly used in time-series forecasting (e.g. [7], [8], [18]). We have built prediction models for the five topmost-starred data visualization projects in GitHub, and assessed the performance of the models for the following research questions:

- RQ1: How does the proposed model predict the number of stars with varying days (k)?
- RQ2: What are the effects of user-score based, weighted metrics on the predicting the number of stars?

Using contemporary forecasting techniques and a combined set of features, we believe our proposed model could guide users to understand the evolution of OSS projects, and to select which projects to contribute in their future development activities [5]. In the rest of the paper, we summarize related work (Section 2), describe our methodology for collecting the projects, extracting metrics and model construction and evaluation (Section 3), discuss our results in Section 4 and finally, we share our conclusions in Section 6.

2 Related Work

As GitHub became the widespread for OSS development, investigating its different aspects has become inevitable for our community. Secondary studies on GitHub (e.g. [13], [9]) report that researchers mine GitHub repositories to understand software development such as pull requests and forks, characterize projects with respect to popularity, collaboration and transparency, and model social factors through users and developers. In predicting the popularity of projects, studies investigate the number of stars, forks, followers and found strong relations among those [9]. Having more consistent documentation as well as having many contributors are found to be other factors explaining popularity of a project [9]. We found many studies on understanding a project's popularity in GitHub, but discussed three studies on *predicting* the popularity of projects in terms of number of stars [3], [4], [17] below.

Borges et al. built models for predicting the number of stars of 2500 GitHub repositories in two of their studies [3] and [4]. In [4], different features from around 2500 GitHub repositories were extracted, such as the project age, the number of forks, commits, contributors, application domain, programming language and repository owner. They analyzed the relationship between these features and the number of stars, and found that there is a weak correlation between the age, the commits, and its star count, however there is a strong correlation between forks and stars [4]. After major releases, projects tend to get a lot of stars. Furthermore, projects are likely to receive more stars right after the first public release. Following that research, [3] expanded their initial model by clustering repositories by their growth rate, making in-cluster predictions and comparing those in-cluster predictions with generic predictions. They built a multiple linear regression model to predict the number of stars of GitHub repositories using previous star counts as their model input. The authors were able to predict the number of stars six months ahead with high confidence using star events of past six months, and they mostly observed more accurate results with in-cluster predictions [3]. Weber and Luo [17] mined two snapshots of GitHub projects, and extracted 38 features, 20 of which were from Python abstract syntax trees (in-code features) and 18 of which were related to project volume, documentation volume, presence of supporting files, and standard library usage. The GitHub snapshots were divided into two even groups, 1000 low popularity and 1000 high popularity projects. Using decision tree classifier, [17] reports the most important features for distinguishing a low popular project from a high popular one as readme file size, continuous integration file size, class definitions, and comment ratios. The division of projects into two classes and the features are different from general practices in this area, and the decision tree classifier is used for finding the importance of features rather than predicting the stars using these features.

In our study, we propose a time-series based model to predict the number of stars received in consecutive k days. Previous works on predicting the number of stars [3], [4], [17] and on predicting the number of forks [6] are similar to our research goal, although they differ from our work in terms of methodology, algorithm and collected data. [4] did not build a predictive model but by using descriptive analytics, it is reported that forks and release dates are important factors for predicting the popularity. We use the forks, commits, releases as some of the features in our model. We also follow prior studies' findings and extend the set of features by incorporating the effect of users (contributors and followers of the projects) [5], [16] and issues [2] on the star counts . [3] uses multi linear regression model to predict star counts, and reported accurate results in most projects, yet considered as unreliable for repositories with non-linear growth by authors themselves. [6] predicts the number of forks after T months since the project's start date and uses a stepwise linear regression model for all projects. Different from both these works, we utilize the power of deep learning methods which do not depend on the linearity constraints. We use a special variation of recurrrent neural networks for time series forecasting as these techniques were

successfully applied to other problems in [7], [18], [12], especially when incorporating many input features in the form of time-series. Furthermore, our model predicts the number of stars received in consecutive k days. We believe predicting cumulative star count is more challenging because it either increases over days or gets stabilized after the project activity degrades. For the projects with high number of stars, the number of stars received over a week/month may not affect the total number of stars and therefore, forecast models may produce low error rates which is misleading for the star count prediction. Therefore, we aim to find the evolution of the number of stars in GitHub projects using recurrent neural networks.

3 Methodology

In this section, we describe our dataset, our methodology of extracting and interpreting metrics, the training algorithm and the model construction.

3.1 Dataset

Our research focuses on a set of interchangeable repositories which operate in the same application area, namely *data visualization*, since the earlier work report that the distribution of the number of stars by application area is significantly different [4]. The dataset used in this study includes historical data of the selected area. All the data was collected through the public API provided by GitHub. For each data-visualization repository, we collected commit, issue opening, issue closing, starring, forking and publishing a release events in terms of who and when performed the event. All these events can be interpreted as potential features for predicting the stars. Commits are the main activities in GitHub. Hence, an actively developed repository with frequent commits may attract the community more than a repository which does not receive commits on a regular basis. Issue tracking systems are significant parts of software development as they lead to the participation of community while reporting bugs and requesting new features [10], [2]. Thus, issue opening / closing activities can be associated with the interest of community. Forks on the other hand, is highly correlated with with the number of stars [4]. Publishing releases also may attract the community. We built a time-series data of each repository. The time-series datasets contain daily number of received stars and other metrics extracted from the events. Even though we collected the whole timeline of the repositories, we limit our dataset to the beginning of 2013. We also selected five topmost starred repositories as our dataset for the experiments. Table 1 shows the selected repositories and their number of stars as of the date their data were crawled from GitHub.

Metrics We have selected six metrics based on the GitHub events collected from data visualization repositories. In order to see the effect of the users, we also derived a time-aware user scoring method, calculated six additional weighted metrics based on user scoring, and computed 12 metrics in total. Since we have

Table 1. Five Topmost Starred Repositories (as of 2017-11-14).

Repository	No. of Stars
d3/d3	70049
chartjs/Chart.js	33416
ecomfe/echarts	21785
Leaflet/Leaflet	19944
gionkunz/chartist-js	10151

Table 2. Metrics and their descriptions.

Metric	Description
Number of Stars	The number of stars received by the repository at day t.
Number of Forks	The number of forks received by the repository at day t.
Number of Commits	The number of commits made on the repository at day t.
Number of Opened Issues	The number of issues opened on the repository at day t.
Number of Closed Issues	The number of issues closed on the repository at day t.
Number of Releases	The number of releases published by the repository at day t.

built a daily time-series data, all of the metrics are computed on a daily basis. The weighted metrics, on the other hand, are aggregated by summing scores of all the users related with the repository until day t. We listed the metrics and their brief descriptions in Table 2.

User Scoring Previous models in the area of measuring developer influence propose metrics like the number of followers in GitHub [10], [2]. [2] also proposed a method for ranking developers through an influence propagation over a network of users. Our scoring mechanism also calculates an influence score for a developer by adhering to the development and community activities within the repositories on data visualization area. However, weighting a metric on day t1 with user score on day t2 may cause incorrect results. Thus, instead of using static user scores, we have created a time-series data of user scores such that the metrics on day t are weighted based on the scores of the users in data-visualization repositories until day t. We have seven main user-scoring metrics. The user-score metrics and their descriptions can be seen in Table 3. Equation 1 shows our heuristic formulation for calculating the user score on day t. We weighted each metric based on its importance value and normalized the metric by dividing it to the sum of the importance values. With this calculation, the impact of a user increases over time as the user actively contributes to the projects in the selected application area.

$$score_{(u,t)} = \frac{\sum_i metric_i(u,t) * importance_i}{\sum_i importance_i} \tag{1}$$

Table 3. User-score metrics, their descriptions and importance (I) values.

Metric	Description	I
Number of Commits	The number of commits made by the user until day t.	7
Number of Closed Issues	The number of issues closed by the user until day t.	6
Number of Releases	The number of releases published by the user until day t.	5
Number of Contributed Repositories	The number of repositories the user made contributed until day t.	4
Number of Opened Issues	The number of issues opened by the user until day t.	3
Number of Forks	The number of repositories forked by the user until day t.	2
Number of Stars	The number of repositories starred by the user until day t.	1

3.2 Algorithm

Even though there are linear regression models and ARIMA to model time series data, we use Long Short Term Memory (LSTM) as our forecast model. LSTM is a special variation of Recurrent Neural Network (RNN) as a solution for the problem of blowing up and vanishing of recurrent back-propagation in traditional RNNs [11]. A simple LSTM unit consists of a memory cell and gates [11]. In memory cell, important information of the input sequence is stored. The gates determine whether the information will be stored or not. Thus, this makes it applicable for predicting time-series. The architecture of LSTM makes it suitable for training steps of time-lag sequences with different input and output steps. Our motivation is to build a flexible forecast model which is capable of making predictions for k consecutive days by learning from historical time-series data, and hence, LSTM fits perfectly to this requirement.

There have been many applications of LSTM for time series forecasting (e.g. [7], [18], [12]). Previous studies show that LSTM outperformed models built with Support Vector Regression on both univariate and multivariate data for phone price prediction in [7], and with ARIMA for CPU usage prediction [12]. It is also proven that multiple features can be fed into time-series forecasting with LSTM [18].

3.3 Model Construction

As LSTM Networks are capable of learning sequences in different fields, this algorithm allow us making multi-step predictions. To accomplish that, we formatted our time-series dataset as input sequences which have historical data and future data consists of each sample corresponding to a day. To answer RQ2, while creating sequences at each time sample, we took data of input metrics in previous 180 days as input, and stars in next 30 days as output. This means that our model predicts stars received daily in the next 30 days given the previous 180 days data. Then, we split the dataset as 70% for training and 30% for test. Instead of a general training model which involves all the data visualization repositories, we have specialized and trained a LSTM Network for each repository individually. These constructed LSTM networks are formed by one input layer, two LSTM layers each of which contain 500 hidden units and an output layer.

3.4 Performance Evaluation

We evaluate our models with Mean Absolute Percentage Error (MAPE) which is widely used for evaluating the accuracy of forecast models. It is defined as follows:

$$MAPE = 100 * (\frac{1}{n} \sum_{t=1}^{n} \frac{|Y_t - F_t|}{Y_t}) \tag{2}$$

where Y_t denotes the actual value, F_t denotes the forecasted value and n denotes the number of samples. Even though MAPE is widely used for evaluating forecast accuracy, outliers drastically increases the error [15]. Therefore we also report the box plots of Absolute Percentage Error (APE) for each model over multiple iterations.

4 Results

We built different models having different combinations of metrics as their inputs. The input combinations are as follows: **Model 1.** Number of stars only. **Model 2.** Repository based metrics (Table 2). **Model 3.** User-Score based, weighted metrics (Metrics in Table 2 weighted based on Equation 1). **Model 4.** All metrics. Our prediction models generated forecasts for the k consecutive days for each instance in our dataset. We chose k to be 1, 7, 14 and 30 indicating predictions made for 1 day ahead, 7 days ahead, 14 days ahead and 30 days ahead, respectively. To answer RQ1 and RQ2, we obtained predictions for consecutive 30 days for each model constituted by different input combinations, and evaluated the input combinations to see how different input metrics impact the performance of the model. Table 4 shows the calculated MAPE values for the input combinations across all repositories.

Table 4. MAPE values for four models across all repositories. The column k indicates the predicted day.

	Model 1				Model 2				Model 3				Model 4			
	k				k				k				k			
Repositories	1	7	14	30	1	7	14	30	1	7	14	30	1	7	14	30
d3/d3	28.56	28.56	28.52	28.76	20.34	20.4	21.06	19.76	20.46	21.26	22	19.79	21.07	21.13	21.23	20.45
chartjs/Chart.js	32.89	32.89	33.56	30.03	33.27	32.2	31.57	37.02	32.09	29.87	29.8	33	31.82	27.93	29.49	31.87
ecomfe/echarts	42.99	42.99	44.05	47.23	39.89	45.07	45.02	48.11	53.13	53.3	53.32	53.67	44.44	50.84	47.72	49.22
Leaflet/Leaflet	35.39	35.39	35.57	37.99	32.95	33.15	33.35	34.62	37.89	37.45	35.56	36.42	36.95	34.69	34.9	36.32
gionkunz/chartist-js	73.25	73.25	69.78	70.48	51	57.31	59.82	66.83	65.98	54.5	58.03	70.57	48.8	48.9	48.85	59.5

RQ1 aims to evaluate the performance of the proposed models on predicting the number of stars for different days. Our experiments were made for k = 1, 7, 14, 30 days. It seems there is not a generic pattern in the change of MAPE values over different k. The models perform almost the same for different k days. There exists an irregular increase in MAPE values for Models 2 to 4, as the predictions are made for further days for *gionkunz/chartist-js*. But the increase is not statistically significant. *d3/d3* is the best performing repository in all cases with MAPE values less than 30%. We also computed the distribution of absolute percentage error (APE) values of the repositories over multiple iterations of model evaluation. Fig. 1 shows APE distribution for *d3/d3* project only, as the pattern is the same for all the projects. For all cases, it is observed that the outlier APE values cause the increase in MAPE.

RQ2 aims to assess the performance of predictions using different metric sets in terms of MAPE values reported in Table 4. For all repositories except *ecomfe/echarts*, adding metrics in addition to number of stars leads to more

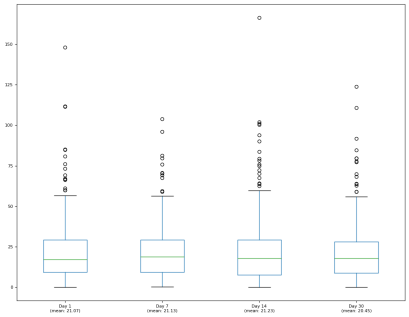

Fig. 1. APE values for Day 1, Day 7, Day 14 and Day 30 for *d3/d3*.

accurate predictions. Especially for *d3/d3* and *gionkunz/chartist-js*, error values significantly decreases when the metrics are included to the model. Even though adding user-score based, weighted metrics slightly decreases the error values for *gionkunz/chartist-js* in all days, and for *d3/d3* for Model 4, repository based metrics (Model 2) or the combination of all (Model 4) perform much better for *d3/d3, chartjs/Chart.js, Leaflet/Leaflet, gionkunz/chartist-js*.

Discussion In RQ1, the performance of the models did not improve as they generated predictions for bigger k days (k=14, k=30). This finding supports the conclusions in [3]: For smaller k values, average relative squared error (RSE) rates are above 30, whereas for k=52 weeks, RSE rates decrease down to 5 in top starred repositories [3]. Among all the models, we achieved MAPE between 19.76 and 70.57 for k=30 days. However, predictions at k=1 day are between 20.4 and 73.25. But the k values picked in our study are still below the k's picked in [3], and hence, it is expected to observe higher error rates than the prior work. Findings of RQ2 also support the previous findings that events in GitHub in terms of development and users contributed to the repositories are good indicators of number of stars. In three of the projects (*d3/d3, ecomfe/echarts, Leaflet/Leaflet*, only repository metrics produce an average MAPE of 20% to 44%, whereas in the two projects (*chartjs/Chart.js, gionkunz/chartist-js*) all metrics perform better with an average MAPE of 30% and 51%. When we look at the projects in detail, each project has a different characteristics and hence, a different best-performing model. The best-performing models are mostly achieved in Model 2 (*d3/d3, ecomfe/echarts, Leaflet/Leaflet*) with varying k. The other two projects (*chartjs/Chart.js, gionkunz/chartist-js*) have better predictions with Model 4 and k=7 and k=14. The deviations in MAPE values show that predictions for recent days are still challenging, and there might be other factors such as social media attention of a project [5] causing sudden increases in star counts for the projects. In order to focus on the area expertise of the developers we only use

their contributions to the data visualization area. However, the features and the LSTM model are independent of the application area. Thus, our method can be applied to different domains.

5 Threats to Validity

In this study we evaluated only one application area in GitHub, namely data visualization, and assessed the performance of deep learning techniques on the prediction of star counts. Fluctuations on growth trends and user interactions particularly in this application area are likely to influence our findings. As this study depends on development activities and community interactions of repositories, our methodology could be applied to other application areas using suitable projects with active development activities and high star counts. Since we used several user-score weighting metrics based on aggregated findings from literature, we had to derive a formula to come up with a single user-score weight at a particular snap shot. The formula used is weighting all metrics based on their importance values decided by votes of the three researchers. These importance values may change, and impact the overall weights of users at any time t, however the values are assigned based on our experience in using GitHub and previous studies' highlights on the important factors influencing a user's impact on the community. We used MAPE to assess the performance of our model as it is stated as a better measure to assess the forecast errors in [15]. We also observed that MAPE is sensitive to outliers, and hence plotted APE values over all iterations of the model.

6 Conclusion

This study proposes a methodology to predict the number of stars received at the consecutive k days in GitHub using LSTM. Our results show that predictions for recent days (up to 30 days) can be made with varying error rates as the performance of LSTM differs among the selected projects. The best performing model is achieved in *d3/d3* project with a MAPE value of 19.76. As a future work, we would like to add user-score metrics using a different heuristics into the model, and observe the effect of these heuristics on the model performance. We would also like to investigate the effects of individual features on predicting the star counts. We believe LSTM performed very well on time-series GitHub events, as it captures changes on development and contributions at the lowest granularity (daily) and adds multiple metrics' time series data into a single model. Therefore it is very powerful for analysis on time-series forecasting problems, and can be configured in various forms in the future. Our proposed LSTM model can also be applied on other repositories.

Acknowledgments This research is supported in part by Scientific Research Projects Division of Istanbul Technical University with project number MGA-

2017-40712 and Scientific and Technological Research Council of Turkey with project number 5170048.

References

1. Badashian, A.S., Stroulia, E.: Measuring user influence in github: the million follower fallacy. In: Int. Works. on CrowdSourcing in Softw. Eng. pp. 15–21 (2016)
2. Bissyande, T.F., Lo, D., Jiang, L., Reveillere, L., Klein, J., Traon, Y.L.: Got issues? who cares about it? a large scale investigation of issue trackers from github. In: Int. Symp. on Softw. Reliability Engineering. pp. 188–197 (2013)
3. Borges, H., Hora, A., Valente, M.T.: Predicting the popularity of github repositories. In: Int. Conf. on Predictive Models and Data Analytics in Soft. Eng. pp. 1–10 (2016)
4. Borges, H., Hora, A., Valente, M.T.: Understanding the factors that impact the popularity of github repositories. In: IEEE Int. Conf. on Softw. Maintenance and Evolution (2016)
5. Borges, H., Valente, M.T.: Whats in a github star? understanding repository starring practices in a social coding platform. Journal of Systems and Software **146**, 112 – 129 (2018)
6. Chen, F., Li, L., Jiang, J., Zhang, L.: Predicting the number of forks for open source software project. In: Int. Works. on Evidential Assessment of Softw. Tech. pp. 40–47 (2014)
7. Chniti, G., Bakir, H., Zaher, H.: E-commerce time series forecasting using lstm neural network and support vector regression. In: Int. Conf. on Big Data and Internet of Thing. pp. 80–84 (2017)
8. Connor, J.T., Martin, R.D., Atlas, L.E.: Recurrent neural networks and robust time series prediction. IEEE Trans. Neural Netw **5**(2), 240–254 (1994)
9. Cosentino, V., Izquierdo, J.L.C., Cabot, J.: A systematic mapping study of software development with github. IEEE Access **5**, 7173–7192 (2017)
10. Grammel, L., Schackmann, H., Schrter, A., Treude, C., Storey, M.A.: Human aspects of software engineering. pp. 1–6 (2010)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computing **9**(8), 1735–1780 (Nov 1997)
12. Janardhanan, D., Barrett, E.: Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models. In: Int. Conf. for Internet Technology and Secured Transactions (2017)
13. Lpez, A.R.: Analyzing GitHub as a Collaborative Software Development Platform: A Systematic Review. Msc thesis, Uni. of Victoria (2017)
14. Neath, K.: Notifications & stars. online (Aug 2012)
15. Shcherbakov, M.V., Brebels, A., Shcherbakova, N.L., Tyukov, A.P., Janovsky, T.A., Kamaev, V.A.: A survey of forecast error measures. World Applied Sciences Journal **24**(24), 171–176 (2013)
16. Tsay, J., Dabbish, L., Herbsleb, J.: Influence of social and technical factors for evaluating contribution in github. In: 36th Int. Conf. on Soft. Eng. pp. 356–366 (2014)
17. Weber, S., Luo, J.: What makes an open source code popular on git hub? In: Int. Conf. on Data Mining Worksh. pp. 851–855 (Dec 2014)
18. Zhang, L., Liu, P., Gulla, J.A.: A neural time series forecasting model for user interests prediction on twitter. In: 25th Conf. on User Modeling, Adaptation and Personalization. p. 397398 (2017)