Analysis of Stack Overflow Interactions

Vasileios Milias, Lorenzo Gasparini, Abigail de la Rosa

I. Overview

Q&A websites started as simple questioning - answering platforms but they evolved to communities which produce and share huge amounts of knowledge. The information that we could derive from the analysis of such websites is valuable for the understanding of the way people interact over specific subjects and for identifying the topics for which people discuss the most.

In our project, we focus on one of the most known Q&A websites: Stack Overflow. We selected 6 different topics, we gathered data of 4 years and we explored their popularity over time. We compared two ways of measuring the popularity: one that it is based on the number of users who have interacted on the topic and another one which is based on the number of the interactions.

In order to measure the popularity, we constructed the question-answer per topic networks. In addition, we examined if other properties of those networks correlate with the popularity of the corresponding topic.

Finally, we were intrigued to test if the various properties of the networks could be proved useful for the time-series prediction of the popularity of each topic. For the evaluation of our predictions, we used the three first years to train our model and we predicted the fourth one. The results vary over the different topics, but the method seems promising and interesting for further analysis.

II. Dataset

We extracted the dataset from the publicly available *Stack Exchange Data Dump*. We considered the *posts* which could be a *question* or an *answer*, of the StackOverflow website. We processed the dataset importing first the XML dumps in a database. Then, we extracted for each topic the question-answer interactions, grouping them by month on the basis of the answer creation date. This resulted in a temporal network for each of the 6 topics with the step size of a month over the years 2013 to 2016. We choose this kind of topics in order to find out the relationship behaviour of the selected programming languages over time.

III. Constructing the network

We explored each dataset with two different approaches. The first approach is user-oriented: we constructed a graph where each node represents a user and each edge represents the interaction of two users (e.g., question - answer).

In this network we are not interested in how many times two users have interacted but only in if they have interacted at least once. Figure 1a shows a simple example of a network where user A has asked a question and user B has replied it.

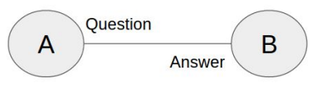

Figure 1a: Network Example

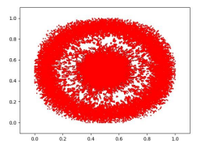

Figure 1b: <json> network (11/2015)

The difference on the second approach is that we are interested in the exact number of interactions. Therefore, we created a multigraph where multiple edges might connect two nodes.

Furthermore, to study the evolution of those networks we splitted our data per month and we examined the temporal networks per each topic. We also created the aggregated graphs, to study the overall relationships per topic.

IV. Measuring Popularity

We decided to measure the popularity in two different ways:

- By counting the number of users who have interacted with a certain topic. In this case, we assume that the fact that the same users talk a lot about a specific topic, does not make the topic more popular. For example, if only two users ask and answer a huge amount of questions it does not mean that the topic is popular. Only the number of people who talk about it, results for the topic to be popular or unpopular.
- By counting the number of interactions between the users. In this case we assume that the popularity is about the amount of questions and answers for each topic.

V. Exploring the overall popularity per topic

Figures 1 and 2 show the overall popularity of each topic (data from 2013 to 2016) by using the two above-mentioned approaches. As we could easily observe, the two approaches give in some occasions different results. Therefore, the way we select to measure popularity leads to a different interpretation of the results.

For example, the *<json>* topic has the largest number of active users and it seems to be the most popular topic in the first approach (Figure 1). However, using the second approach we discovered that the number of interactions on the *<angularjs>* topic is higher. Thus, while a lot more users have questioned or answered about *<json>* than *<angularjs>*, more questions or answers have been posted about the latter. For *<json>* there is a larger community which is interested about the topic, but for *<angularjs>* the community is more active. Those numbers are also influenced by the nature of the topic itself.

Nevertheless, since *<json>* is an older topic than *<angularjs>*, various questions have already been answered and the users are able to find answers to their questions without posting new ones. This, does not make the topic unpopular. That's why we support that by using both these metrics is a good way to understand the popularity of a topic and using only one of them is insufficient.

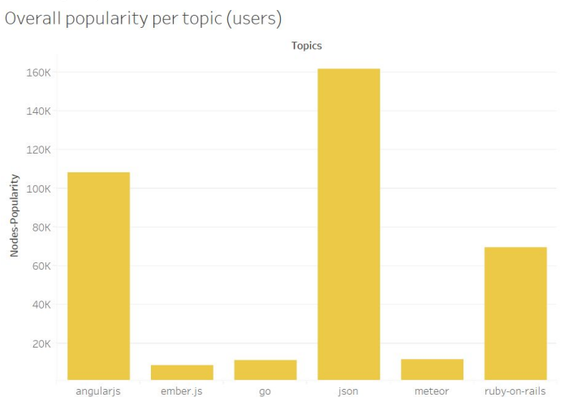

Figure 2

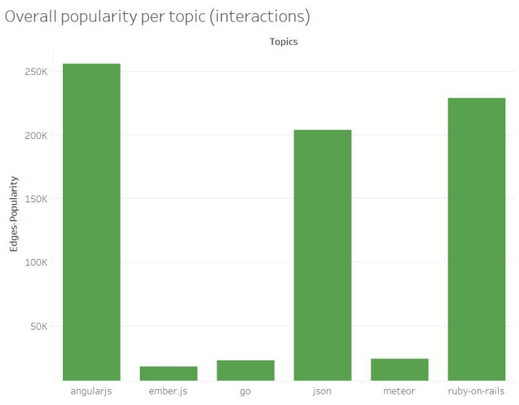

Figure 3

VI. Exploring the temporal popularity per topic

To further investigate the topics' popularity we calculated the two different popularity metrics per month. In Figure 4 we present the topics: *<angularjs>*, *<ruby-on-rails>*, *<json>* and *<go>*. The green line shows the number of interactions per month and the orange one shows the number of users. As expected, the number of users is directly correlated with the number of interactions and the general trend of both lines is the same for all the topics.

Our results come to an agreement with the results of http://sotagtrends.com/, where they use another metric to calculate popularity: number of questions. In addition, we tried to connect our results with our prior knowledge about those topics.

Our results, suggest that **AngularJS** started losing its popularity from March of 2016. Indeed, the rewrite of AngularJS, named simply Angular, had its first release candidate on May of 2016. As expected, more and more people started using Angular, which uses the *<angular>* topic and therefore *<angularjs>* got less and less popular.

JSON's overall trend shows that its popularity increases throughout the years. This is something that we also expected as the JSON format in the recent years has taken the place of XML as data exchange format. The number of users of the *<json>* topic is more than the number of interactions. Thus, there are not so many interactions between the same users and there are possibly various unanswered questions.

The **Ruby on Rails** framework seems to have lost popularity over the past few years. Indeed, while it was very popular once, the past 4 years Node.js is being used more and more and it basically has outran Ruby on Rails.

Go is a relatively new programming language created in 2007 by Google. One of the main features of this programming language is the ability to support networking and multiprocessing. In 2016, Go's popularity skyrocketed because of the rising importance of Docker and Kubernetes. The *<go>*-graph depicts this information. In general, Go's popularity increased throughout the years as it is shown in our results.

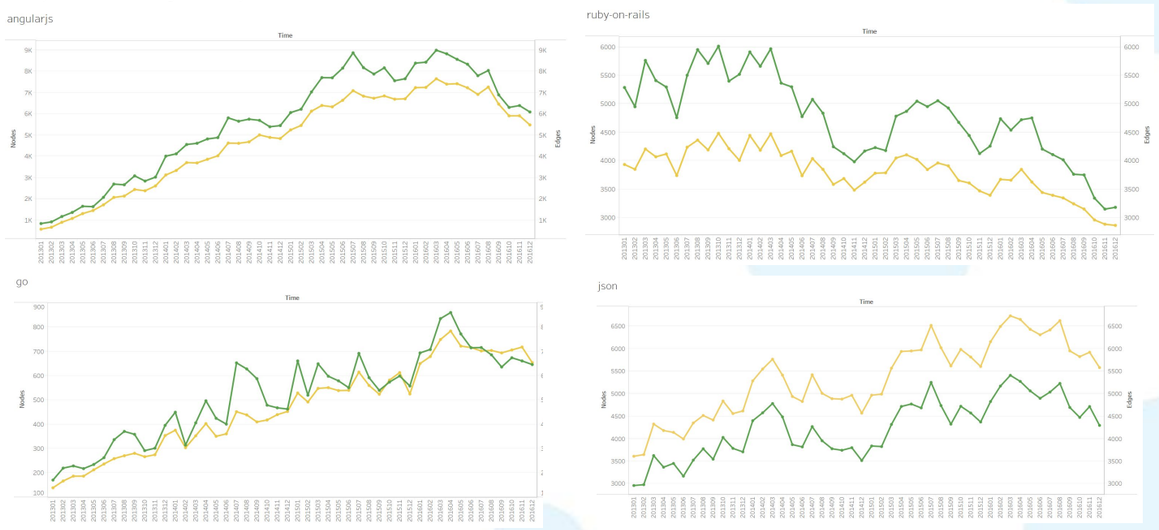

Figure 4: *Number of interactions* *Number of users*

VII. Exploring other metrics of the temporal networks per topic

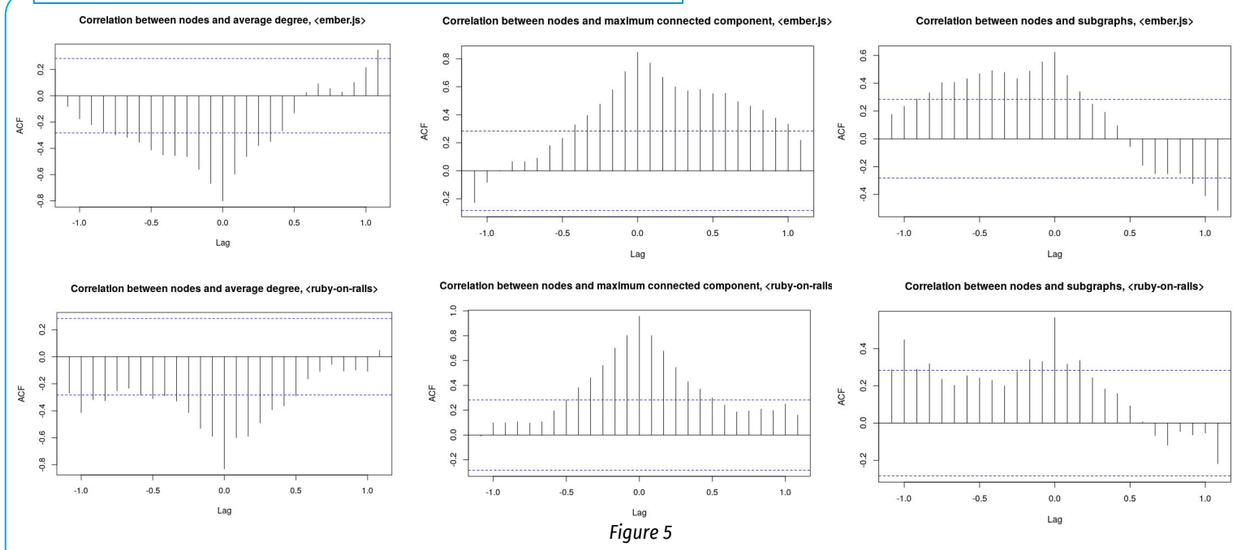

Figure 5

Each topic evolves differently throughout the years. We strongly believe that the network coefficients of those constantly and dynamically changing networks could help us better understand the nature of each topic and provide us with insights. By using cross correlation we computed the correlation between the temporal popularity (measured as the number of users) and the three following metrics: the **average degree coefficient**, the number of connected components (**subgraphs**) and the **size of the maximum connected component** (number of users of the largest subgraph).

In Figure 5 we present our results for two of the above-mentioned topics: *<ember.js>* and *<ruby-on-rails>*. The *y*-axis shows the correlation coefficient between a specific metric and the popularity (number of users) and the *x*-axis shows the lag in terms of years.

As it can be observed, the number of subgraphs in the present is highly correlated with the popularity (number of nodes) in the future (up to lag 1 year). In platforms such as Stack Overflow the users try to reply the most difficult answers to get a higher score. The communities that are being created are somehow per level. When the number of communities is small it is more difficult for a new user to find the communities which match her own level and ask or answer questions. Therefore, it is more difficult for a new user to be part of a topic. However, as the number of subgraphs increases more people are becoming able to join (positive correlation). Thus, the number of subgraphs looks like an informative metric for the evolution of a topic.

As expected, the average degree coefficient shows a negative correlation for values of the lag close to 0. Obviously, as the popularity of a topic increases each user is directly interacting with a smaller percentage of the overall users.

Finally, the size of the maximum connected component shows also positive correlation for positive values for the lag, indicating that an increase in the popularity in the present is associated with an increase of the size of the maximum connected component in the near future.

VIII. Predicting the popularity

In the final part of our research, we studied ways to predict the popularity of a topic in the future, using time-series forecasting. We wanted to test our intuition that incorporating the characteristics of the temporal network in the model could indeed improve the predictions.

The goal was to predict the popularity of each topic in 2016, in terms of nodes in the network, using the popularity over the years 2013-2015. We compared two scenarios: in the first one the prediction was made using only the historical values of the popularity, while in the second one we also incorporated in the model the historical values of the following metrics:

- Average degree
- Clustering coefficient
- Number of subgraphs
- Number of nodes of the largest connected component

For the prediction we used the R package *forecast* and in particular the *nnetar* function, which does univariate time series forecasting with a feed-forward neural network with a single hidden layer and lagged inputs. In particular, a **NNAR(*p*,*P*,*k*)[*m*]** model is fitted, using the lagged values of the time series, and eventually other time series, as input and a single hidden layer with *k* nodes (half of the number of input nodes plus 1). The inputs are for lags 1 to *p* and *m* to *mP*, where *m = 12* in our case, to take into account the seasonality of the data. *P* is selected to be 1 to take into account only the last value of the previous season and *p* is automatically selected from the optimal linear **AR(*p*)** model fitted to the seasonally adjusted data. The neural network is fitted for 20 times with different starting weights and the predictions are then averaged.

We did the forecasting on each of the 6 topics and we measured the quality of the prediction using the mean squared error between the observed values for 2016 and the predicted ones.

Tag	MSE not using network metrics	MSE using network metrics	Difference
angularjs	468020	500956	+7.04%
ember.js	7355	1971	-73.2%
go	10755	11002	+2.3%
json	1261785	482568	-61.7%
meteor	39669	35359	-10.8%
ruby-on-rails	117165	103905	-11.3%

Table 1

In Table 1 we list the results of the prediction of the popularity in the two scenarios. It can be seen that for most of the topics incorporating the network metrics results in a significant decrease of the mean squared error, except for *<angularjs>* and *<go>*, which see an increase of 7.04% and 2.3% respectively.

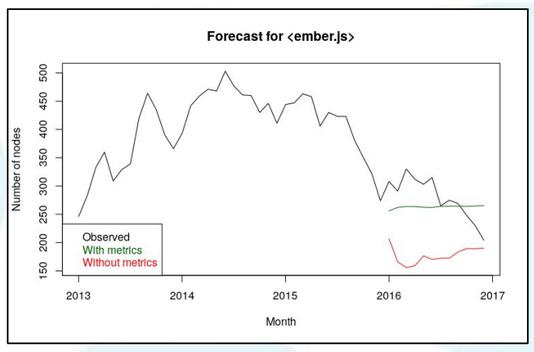

Figure 6

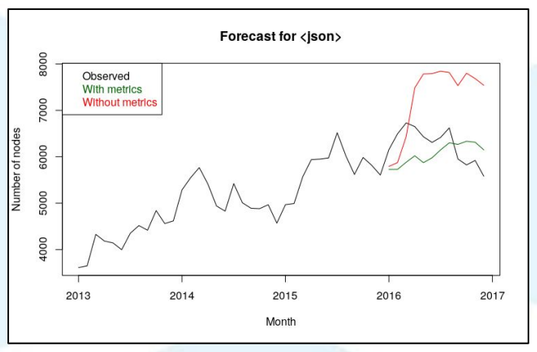

Figure 7

In Figure 5 and 6 the predictions for the *<ember.js>* and *<json>* tags are visualized. In these two cases, as can be observed in the figures, the historical metrics of the networks improved the predictions.

IX. Conclusions

In this project we studied the evolution of popularity in Stack Overflow. We explored two ways of measuring the popularity on a topic-level. To further understand the differences between our two metrics we investigated both the static overall topic-networks and the temporal networks (on a per month basis). We combined our results with the primary knowledge that we had for the popularity of those topics and we showed that even if both our metrics could be well justified, they might end-up in different results.

In addition, we examined other metrics of the temporal networks we constructed, and we discussed how and why they correlate or not with the popularity.

Finally, as part of an exploratory analysis, we attempted to predict the popularity of each topic based on its history. We created and compared two prediction models: one that uses only the popularity history and another one which also uses various metrics of the temporal networks. The use of the latter model improved our predictions, and we think it is worth to further investigate this area.

DELFT DATA SCIENCE

EE4389
Modeling and Data Analysis in Complex Networks

TU Delft
Technische Universiteit Delft