AN INTERACTIVE VISUAL APPLICATION TO EXPLORE COMMUNITIES ON STACKOVERFLOW

Amos Tan Wei Jie, Ong Sue Cern, Tay Wei Rong

Abstract **–** Stackoverflow is a growing questions-and-answers site for software programming. Over the 9 years of the site, it has collected a vast amount of rich and helpful data, contributed by its active community of users. This research project aims to understand interactions between Stackoverflow users and how they use various technologies. To achieve this, our team designed an interactive visual application utilizing data from Stackoverflow's data dump. Visualizations for the application make use of sigma.js and D3.js to display graph networks. Through this data visualisation application, our team hopes to help both users and Stackoverflow understand user interactions and technology trends on the site better.

I. INTRODUCTION

Founded in 2008, Stackoverflow is the flagship site of the Stack Exchange Network, which features questions and answers on a wide range of topics. Stackoverflow focuses solely on software programming questions, ranging many technologies.

Users can ask and answer questions, and be involved through commenting, voting and editing posts. Based on their activity, they can earn reputation points and badges, thus indicating high-value posts and contribution.
Stackoverflow maintains quality posts through self-regulation by the community. Its success, reaching over 50 million people globally[i] each month, has led it to expand into partnering with employers for recruitment of developers. It also provides business solutions to businesses based on insights from its sites. With a rich amount of data available, we are interested in how the community has evolved over time for the different technologies, and how this reflects different trends over time.

II. MOTIVATION

Our research is motivated by the lack of information on community interaction within Stackoverflow. As Information Systems students who view the site frequently and rely heavily on it for assistance, we realised that we are surprisingly unaware of how interaction occurs on the site to keep the community active.

Most visualizations available show the growth and popularity of technologies, but not the underlying users driving these trends. We find it important to study user-to-user interaction as the community plays a very important role in providing useful content to the website.

Apart from the users, the use of multiple tags of technologies (e.g. javascript, jquery, react-native) which are placed on a question posted allow us to study how closely related different technologies are to one another. This is useful in understanding ways to integrate separate technologies with one another, and to gauge the popularity of using these technologies together.

III. RELATED WORKS

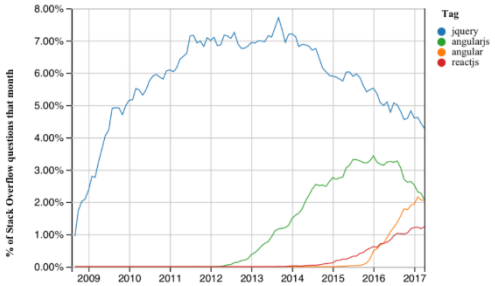

Fig 1: Interactive chart showing % of Stackoverflow questions per month by tag, Stackoverflow Insights Website[ii]

Stackoverflow has published insights, mainly on the popularity of different technologies over

time, based on questions asked on its website. As this is plotted in a time series and in comparison with other technologies, this allows viewers to quickly grasp which technology is gaining traction and support on the site. However, there is a lack of visibility into changes in user interactions and contributions over time, as technology trends change. We believe that this is important, in order to understand how Stackoverflow contributors assist users, once users are interested in picking up the technology. Identifying valuable users within the community would also help Stackoverflow and related technology companies hire talent or increase influence of Stackoverflow.

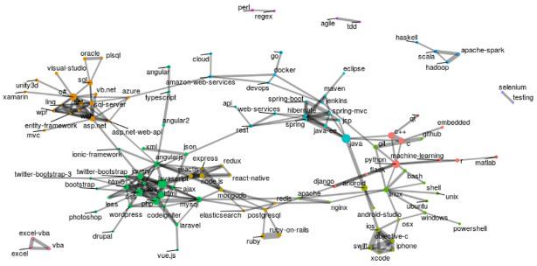

Fig 2: Graph network showing relation of tags on Stackoverflow, Public Kaggle Project[iii]

Another related work found was a network graph on the tags used on Stackoverflow and their relations to one another. The visualisation provides a clear understanding on how the different technologies on Stackoverflow are related to one another. Various technologies are coloured differently to indicate that they belong to separate groups. The length of the edges relays the closeness of a technology to another, and node size likely translates to the popularity of the technology. However, it is hard to notice patterns in the graph as the clusters are not very prominent. The graph is cluttered and static, so it is hard to zoom in to understand each cluster better and to rearrange nodes and edges for a better view. Overlaps between different clusters are also hard to identify. We believe that such a visualization will be more suitable on d3.js.

IV. VISUALIZATION APPROACH

Our group extracted the Stackoverflow raw data from the Stack Exchange Data Dump, and used all the data from 2008 to September 2017, as last updated in the data dump. The files, in XML format, consist of badges, posts, comments, post links, users and votes. We excluded post history as the data was too focused on individual changes made to a posts, rather than general patterns of the community. Based on the different raw files, we came up with a data model to describe interactions on the site by users.

Fig 3: Data model showing the interactions between nodes

The total size of the files when uploaded was 87.71GB, consisting of 178 million nodes and 401 million edges. Given the large data size and processing power required, we hosted the database on Google Cloud Platform, using a 32GB RAM, 6vCPU server.

Our group used a graph database, neo4j to store our data. This allowed us to query on the database browser interface and thus, explore our data easily. Using the Cypher query language, we could specify relationships and nodes to be returned. Query results would be returned in a graph showing interactions among the different type of nodes. However, due to the large dataset, we faced challenges in returning query results for large queries e.g. returning questions posted by tags, sorted on creation date. With limited RAM, both the database and browser crashed multiple times. To prevent this, we indexed our database on certain attributes and made more specific queries.

In order to make sense of the data, we used Gephi, a graph visualization and exploration offline platform, to run analysis on the graphs. We used the neo4j plugin available on Gephi to import the nodes and edges data, which would be

streamed into Gephi. The neo4j database also comes with a library, APOC, that allows returned nodes and edges, specified in the query, to be streamed to Gephi.

Iteration 1 – Non-graph visualizations
Initially, our group was interested in predicting speed of response to a question as we felt that it would be a good incentive for viewers on the site who were hesitant to post questions. We also wanted to visualize top contributors by measuring badges, posts, aggregated upvotes and years of experience. However, we realised that such visualizations do not help us to understand community interactions better, hence it was not in our interest

Iteration 2 – Mixed-type nodes in graph
Our group wanted to visualise in a graph the interaction between questioners, answerers and posts to reflect the interaction among users in a community. However, having 2 different types of nodes: users and posts made it difficult to run analysis on the graph and to understand patterns better. The use of mixed-type nodes in the graph is limited as patterns between graphs can only be easily observed through 'eyeballing'. Moreover, they are only comparable if graphs of 2 different technologies have similar network size, which greatly limits its use.

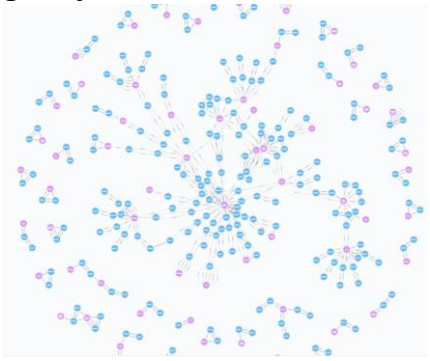

Fig 4: Mixed-type nodes graph with users and posts as node types.

Another graph we considered were post hyperlinks between various posts. With a vast amount of data available, our group wanted to visualise the two main types of post links: duplicate questions and references to other posts (similar to citation/PageRank algorithm). However, this was less suited in helping us understand interactions between users. Instead, it would be more useful to conduct text analytics on these posts to understand what type of questions are commonly referenced or repeatedly asked.

Iteration 3 – Filters on Graph
We wanted to use cross-filter to slice and filter data on attributes such as time and country. However, we dropped the country filter as it would result in more disconnections on a graph network. It would be more suitable to plot growth of new technologies and their spread across countries using other types of visualizations e.g. map, however due to time constraints, we were unable to implement this idea.

V. VISUALIZATIONS SELECTED

Visualization 1: Cluster of Tags
First, we queried our neo4j database on the frequency of 2 tags being used together in a post and plotted the streamed data in a graph on Gephi.

To calculate the various clusters, our group used Gephi's modularity analysis, which makes use of the Louvain method for community detection. The Louvain method uses the 'greedy algorithm' to assign nodes to communities, then evaluate how densely connected the nodes are in the community as compared to average closeness if they were arranged randomly, which is known as the graph's modularity. It fine tunes the assignments until it reaches the maximum modularity possible. The Louvain method is superior due to its fast processing for large graph networks.[iv] When running the graph of tags, a modularity of 0.631 was achieved for the entire network, with 12 communities identified.

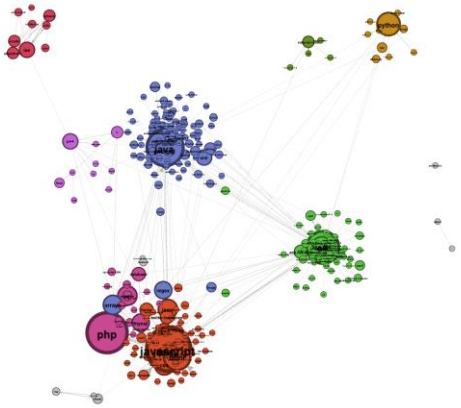

Fig 5: Clusters of tags on Gephi, with data streamed from neo4j database

We then exported the nodes and edges CSV files from Gephi to be used in our D3.js visualization on our application. We selected D3.js to visualize the graph as it allows for interactivity on rearranging the positions of nodes and edges by dragging the nodes. The graph was displayed using a force-directed layout available on D3.js. We also used convex hulls on the various clusters, which outline the area of a cluster. This allows viewers to identify the clusters easily and make inferences on the size and overlaps of each cluster. The convex hulls make the graph simpler to grasp and more aesthetically pleasing. In addition, we added a filter by cluster to allow users to interact with the different clusters and view changes to the graph once clusters are removed.

Visualization 2: Network Graph Analysis

To carry out analysis between users of Stackoverflow, our group had to link users to one another by proxy. This meant that the link from a user to another user is indirectly formed via an activity on a creator's post. Hence, to draw the graph, we had to summarise activities between users as answering a question, commenting or editing on another users' post.

We wanted the search to be carried out dynamically based on the tag entered. This would allow for flexibility in searching on the hundreds of tags available on Stackoverflow. When the graph of user-to-user interaction is returned, by default, the nodes would be populated randomly, thus overlapping one another and making it hard to make sense of any pattern. Hence, we used sigma.js, a graph-drawing library with the ability to run force-directed layout animations on large graphs in order to spot node connections easily in the graph. Viewers can also click on a specific node and to highlight the node, its edges and neighbouring nodes, allowing viewers to observe the connections of a node easily.

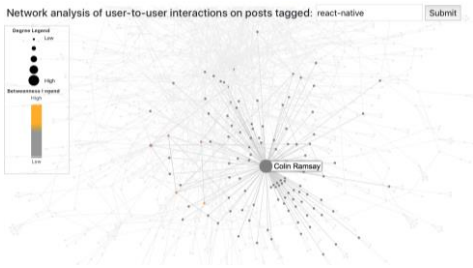

Fig 6: Clicking on a node allows highlighting of its neighbours and connections

We measured the degree centrality of each user based on the in-degree and out-degree edges, which represent interaction. Out-degree centrality allows us to measure activity of a particular user in the network in reaching out to different posts. On the other hand, in-degree measures high attention received by posts of a particular user. We also measured the betweenness centrality of each user, which indicates the amount of information flow in the network that will pass through the user. This is calculated by summing up the number of shortest paths from any node in the network to another that passes through the node measured, divided by the total number of shortest paths in the whole network.

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

Fig 7: Mathematical formula for calculating the betweenness centrality of a node v, by taking the summation of sets where shortest-path (s-t) passes through v, out of all paths[v]

Betweenness and degree centrality are calculated per user based on the technology or tag only, hence a user which belongs to many different technology communities would have different centrality scores for each community.

VI. KEY FINDINGS & INSIGHTS

Cluster of Tags

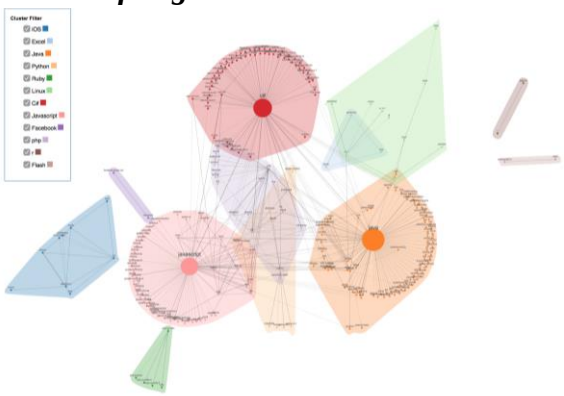

Fig 8: Clusters of tags visualized with D3.js for interactivity

Based on the undirected graph visualized, we notice the 3 largest clusters of tags: C#, Javascript and Java. The cluster size indicates many related tags, which are mainly popular tools and libraries. We observe that for Javascript, the most frequent tag is jQuery and JSON as the edges between them and Javascript is very short. The most frequent tag for C# is ASP.NET and

ASP.NET-MVC while for Java it is arrays, Android and regex. We also observe that Javascript is very versatile and bridges 3 other clusters, Ruby, Facebook and iOS to the main network.

Some of the tags returned are the popular features of a technology such as dictionary and list for Python; arrays, regex, for-loop and collections for Java; events for C# and callback and promise for Javascript. Due to the specific nature of tags, we can thus observe that these are the most asked functions of the languages.

Another cluster that is fairly large would be the light purple cluster with PHP and the various databases. After Javascript, Java and C#, PHP is the 4th highest in the number of ties with other nodes. It has direct links with Java and Javascript, as well as other tags in both the Java and Javascript clusters.

We observe that the light green cluster with Linux, C and C++ is quite sparse with few connections. It is also interesting to note that R, a popular programming language for data analytics, is used mainly in isolation by Stackoverflow users, as it is not connected to the main network. Another cluster in isolation to the main network is Flash, which has been declining in popularity.

Network Analysis

To understand each community of tags better, we conducted network analysis on user-to-user interactions within a community. Using a brush on a timeline to select a time period, the user interactions within the period is returned in the graph. The timeline includes a bar graph of interaction count for each month, so users can identify the trend in popularity of the particular technology, or identify when it was introduced (if after 2009).

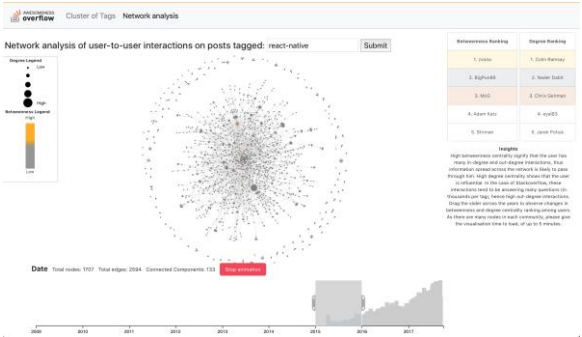

Fig 9: Visualization of react-native community in 2015.

Based on Fig. 9, the visualization for the 'react-native' community is shown. The year specified in the timeline at the bottom is 2015, which is the year it was introduced. Nodes are sized based on out-degree, while their colour is determined by their betweenness score. Top 5 users in both categories are listed in a table on the right so that viewers can observe changes in top contributor rankings over time. To understand who these top contributors are, clicking their name in the table brings the viewer to view their user page on Stackoverflow. By comparing the graph for a technology over time, we observed several patterns.

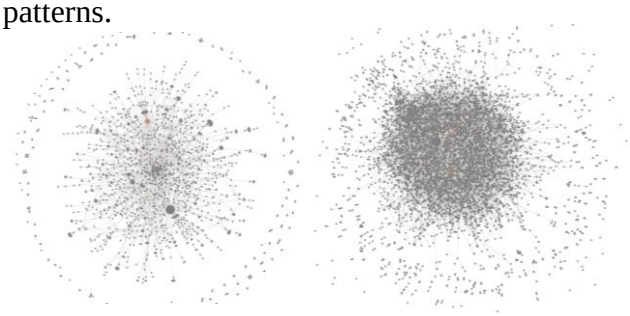

Fig 10: React-native community in 2015(left), which was its first year of launch, and in 2016 (right).

From 2015-2016, the number of nodes increased by 4 times, while the number of connected components (lower value indicates higher connectedness) increased by 3.9 times. This shows us that even though the graph looks much more connected in 2016, it is only slightly more connected.

Betweenness Ranking	Degree Ranking	Betweenness Ranking	Degree Ranking
1. zvona	1. Colin Ramsay	1. vijayst	1. Nader Dabit
2. BigPun86	2. Nader Dabit	2. Coyote	2. Cherniv
3. McG	3. Chris Geirman	3. Chris Geirman	3. Jickson
4. Adam Katz	4. eyal83	4. Rajesh	4. agent_hunt
5. Stirman	5. Jarek Potiuk	5. user2433617	5. G. Hamaide

Fig 11: Top contributors in react-native community in 2015(left) and in 2016 (right).

From 2015-2016, the top contributors are all different, except for Nader Dabit[vi], who moved from 2nd to 1st place in degree ranking. When we click on his name, it shows us his current profile of top tags in 'react-native', asking a total of 8 questions and answering 208, hence the high out-degree score. Looking at vijayst's profile[vii], the top in betweenness ranking for 2016, we notice that he has a more balanced proportion of

questions asked and answered, which are 14 and 133 respectively. Hence, his high betweenness score indicate that information flowing from one node in the network to another is likely to pass through him.

We can also compare connectedness between different technologies.

Fig 12: React-native community in 2015(left), which was its first year of launch; and ionic-framework community in 2014 (right), also its first year of launch.

For both 'react-native' and 'ionic-framework' in their first year of launch, we notice that their network size is very similar of 1707 nodes for react-native and 1751 nodes for ionic-framework. However, their connectedness and density differ quite significantly. React-native has more edges (2594) as compared to ionic (2203), despite having fewer nodes, hence react-native community is a denser network. Based on both the number of nodes and connected components, we observe a rough estimate of about 13 nodes in one connected component for react-native, as compared to 9 nodes for ionic-framework, indicating that react-native community is more connected.

In the case of react-native and ionic-framework, the connectedness of the community have increased as popularity increased. Hence, we studied if this was also the case for a technology with falling popularity. Using Silverlight, as the community declined in numbers, its connectedness was severely reduced. The estimated number of users per connected component dropped from 13 users to 3 users from its peak in 2011 to 2016 levels. A similar trend is noticed in Flash, where users per connected component dropped from 9 to 5. This shows that a large decline in a technology community results in an increase in sparse one-off interactions and questions.

Fig 13: Silverlight community in 2011 (left), when popularity peaked; and in 2016 (right), in decline.

VII. CONCLUSION

Stackoverflow consists of various communities, which span a wide range of languages, libraries and tools. Understanding how different group of technologies are used together enables users and Stackoverflow to recommend new ways to integrate different technologies with one another. By understanding which contributors are essential in the network due to high contributions in answering and in passing of information across the network, Stackoverflow can look into more ways to encourage such quality engagement.

---

[i] About. (n.d.). Retrieved November 25, 2017, from https://stackoverflow.com/company

[ii] Robinson, D. (2017, May 9). Introducing Stack Overflow Trends. Retrieved November 25, 2017, from https://stackoverflow.blog/2017/05/09/introducing-stack-overflow-trends/

[iii] Silge, J. (n.d.). Network graph. Retrieved November 25, 2017, from https://www.kaggle.com/juliasilge/network-graph/code

[iv] Blondel, V. D., Guillaume, J., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment,2008(10). doi:10.1088/1742-5468/2008/10/p10008

[v] Ulrik Brandes: On Variants of Shortest-Path Betweenness Centrality and their Generic Computation. Social Networks 30(2):136-145, 2008. http://www.inf.uni-konstanz.de/algo/publications/b-vspbc-08.pdf

[vi] User Nader Dabit. (n.d.). Retrieved November 25, 2017, from https://stackoverflow.com/users/2864119/nader-dabit

[vii] User vijayst. (n.d.). Retrieved November 25, 2017, from https://stackoverflow.com/users/558972/vijayst