VisUnit: Evaluating Interactive Graph Visualizations Using Crowdsourcing

Mershack Okoe* Radu Jianu

Florida International University

ABSTRACT

The process of evaluating visualizations can be time-consuming. Here, we present a design aimed at automating the process of performing quantitative and qualitative evaluations of graph visualizations by leveraging crowdsourcing, and a set of predefined evaluation modules based on a graph task taxonomy. Specifically, we allow designers to quickly set up a user study with representative graph tasks, measurable metrics, and evaluation methods. Our system then uses a thin-client architecture to automatically generate a web accessible user study from our desktop visualization, places the study on Mechanical Turk, and uses a statistical package to automatically process incoming results. To evaluate our system, we performed three concrete evaluation studies, all of which were configured and deployed in less than an hour. We discuss how our system can be used for automatic evaluations of interactive graph visualizations, how it can facilitate evaluation of alternative designs during iterative design processes, and how it could be used to find good default configurations for graph visualizations.

Keywords: Graph evaluation, crowdsourcing graph evaluations.

1 INTRODUCTION

We explored a design aimed at automating controlled evaluation studies of interactive graph visualizations by leveraging the Mechanical Turk crowdsourcing platform. We evaluate it, and show its potential for getting quick feedback on graph designs. Performing evaluation studies for visualizations can be a tedious and time-consuming task. Crowdsourcing has been shown to be a valid platform for performing visualization experiments [2, 3]. Compared to lab-based studies, crowdsourced studies provide easy access to a more diverse population and high number of participants [3]. In this work, we show that crowdsourcing can be leveraged to automate the evaluation of interactive graph visualizations. This way, evaluation studies could be performed more frequently to guide the design of graph visualizations, for instance between stages of iterative development.

Specifically, we allow a graph visualization developer to quickly set up a user study with representative graph tasks selected from the graph task taxonomy of Lee et al. [4]. We automatically generate a web-accessible user study of the graph visualization, place the study on Mechanical Turk, and evaluate the results for the developer automatically using statistical measures that are aligned with the user study design. We used a thin-client architecture to create web-accessible content from our desktop based system.

Our contribution lies in introducing a design that leverages crowdsourcing to simplify the evaluation of interactive graph visualizations. To demonstrate the potential of the approach, we performed three concrete evaluations all of which were configured and deployed in less than an hour. Results from one of the studies showed that users had better accuracy with interactive graphs compared to static graphs.

* email:{mokoe001@cis.fiu.edu, rdjianu@cis.fiu.edu}

2 RELATED WORK

In visualization, crowdsourcing has been shown to be a good platform for performing evaluation studies. Notable evaluation studies include works of Heer et al. [2], and Kosara et al. [3]. These studies were specific and manually set up.

Our design is aimed at automating controlled evaluation studies of static and interactive graph visualizations by leveraging crowdsourcing. Our implementation leverages a desktop visualization and thus uses a thin-client architecture to deploy the visualization to the web, but the principles of our design could easily be extended to web visualizations.

Our work is most similar to efforts on simplifying the design of controlled experiments. TouchStone [6] is a platform for designing lab-based controlled HCI experiments, and EvalBench[1] is a software library that supports lab-based evaluation studies in visualization. Our work was also inspired by TurkIt[5], a toolkit that leverages crowdsourcing for iterative text editing tasks, but their approach is more automatic than ours.

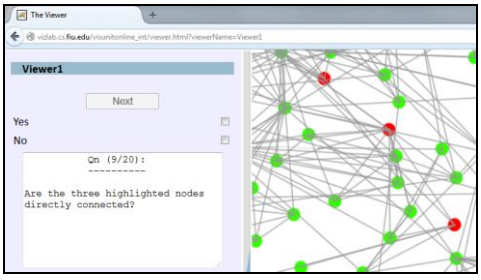

Figure 1: An example of a task involving three nodes.

3 METHODS

First, to simplify graph visualization studies, we provide an interface where designers can configure the settings for the evaluation designs. Such settings include: the graph visualizations for the control and test conditions, whether the evaluation design is within or between users, and the quantitative and qualitative questions to be included in the study. We provided representative tasks from the task taxonomy of Lee et al. [4]. The tasks include topology tasks (e.g. Are two highlighted nodes connected? Is there a path between two or three highlighted nodes?), attribute tasks (e.g. Is there an adjacent node starting with a letter?), and browsing tasks (e.g. Find the number of nodes on a given path that starts with a letter). The designer can also select qualitative questions (e.g. rate the difficulty of the visualization) or define qualitative questions of their own. Additional configurations for the evaluation include specifying the number of assignments to generate on Mechanical Turk, the designer's Mechanical Turk access information, and the reward to

be given to users. On a click of a button, the HIT is created on Mechanical Turk, and workers accepting the HIT can perform the tasks on the webserver where the visualization is hosted.

Second, because our framework leverages a desktop visualization development framework, we ensured that graph visualizations developed on the framework can be made web accessible through a thin-client architecture. Thus, when the visualization is accessed on the web, all rendering of the visualization occurs on our servers and images are sent to the client. Our thin-client achieves up to 10fps making it easy for users to interact with the visualization without lag. However, we note that our design principles would work just as well with web visualizations created in D3, provided an interface mechanism was developed.

Third, we used methods of designing evaluation studies from experimental research. As a default dataset we used a book recommendation network (900 nodes, 2500 edges). We created sets of target nodes for each task type (e.g., pairs of nodes for connection or path tasks). Depending on the number of questions requested by the developer, target nodes are selected from these predefined groups and highlighted in the visualization when the tasks are presented to the worker. Also, when nodes are highlighted in the visualization as part of a task, the visualization is automatically centered so that those nodes are in view. When the evaluation is started, the user is given an instruction about the types of tasks involved in the study. The user is also guided through a training session involving 2 questions for each type of task involved in the study. During the training session users are told whether their answers are correct or not.

In between-user studies, half of the subjects solve the tasks in the control condition and half in test condition. In a within-user study all subjects solve the tasks twice, once in the control condition and once in the test condition. Our design automatically accounts for learning effects by starting half of the users in the control condition and the other half in the test condition.

For each user and each task our implementation records time and accuracy. Results are saved on the server and upon request they are automatically downloaded and analyzed using the R statistical package. So far, we generate boxplots that compare the average time and accuracy on tasks for the control and test conditions and we generate text files containing the results of statistical analyses. These analyses are correlated with the selected user study design: for within user designs a Shapiro-Wilk analysis tests for normality and is followed by a paired t-test, while for between user designs a Kruskal-Wallis analysis is performed.

4 EVALUATION

To evaluate our system, we performed three evaluation studies. The first evaluation was a between user study involving 40 MTurk workers. Half of the workers performed the tasks with interactivity disabled (control condition), and the other half performed the tasks with interactivity enabled (test-condition). We used two main tasks: determine whether two highlighted nodes are directly connected? (12 questions), and determine whether there is a direct path between 3 highlighted nodes (8 questions). The graph visualization used was drawn with the neato algorithm, and we use a book recommendation dataset (900 nodes, 2500 edges). We measured time and accuracy. Results from this study showed that users had better accuracy with interactivity enabled than with interactivity disabled as shown in Fig 1(a). The other two evaluations we performed were within user studies involving 30 participants each, and were used to test how changes in two graph readability metrics (i.e. node-color, and edge-size) affect task accuracy. The second study evaluated node-color (green vs. yellow), and the third study evaluated edge stroke size (2 vs. 6). For the second and third studies, the questions were similar to the first study, and users performed the tasks with interactivity enabled. These three studies were set up in less than an hour and analysed with a click of a button when tasks were completed by MTurk workers. The results from the second and third studies did not show any significant difference in user performance.

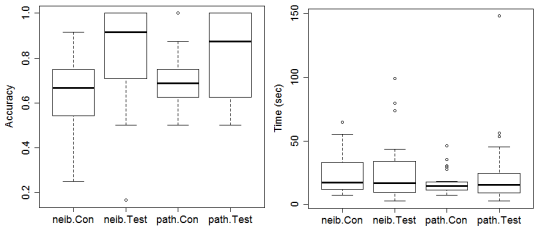

Figure 2: Boxplots results for the interactive vs. non-interactive visualization evaluation (where ".Con" and ".Test" refers to the control and test conditions respectively.

5 CONCLUSION

We explored a design aimed at automating controlled evaluation studies of interactive graph visualizations by leveraging the Mechanical Turk crowdsourcing platform. We provide an interface where a designer can quickly set up a user study with representative graph tasks, automatically place the study on Mechanical Turk, and evaluate the study results using the R-statistical package. Results from our user studies shows that such a system can help designers gain quick feedback on their graph designs. Our current work presents a design and evaluation. Additional work is needed to transform this design into a useable testing platform. Future work will include: multiple datasets representative of diverse graph topologies and sizes; more comprehensive data reporting (e.g. bar charts, and other statistical analysis methods such as Mann-Whitney and ANOVA); and a way of connecting the evaluation engine to third party web-visualizations. However, our design represents a first step in automating the process of evaluating visualizations.

REFERENCES

[1] Wolfgang Aigner, Stephan Hoffmann, and Alexander Rind. Evalbench: a software library for visualization evaluation. In *Computer Graphics Forum*, volume 32, pages 41–50. Wiley Online Library, 2013.

[2] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 203–212. ACM, 2010.

[3] Robert Kosara and Caroline Ziemkiewicz. Do mechanical turks dream of square pie charts? In *Proceedings of the 3rd BELIV'10 Workshop: BEyond time and errors: novel evaLuation methods for Information Visualization*, pages 63–70. ACM, 2010.

[4] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pages 1–5. ACM, 2006.

[5] Greg Little, Lydia B Chilton, Max Goldman, and Robert C Miller. Turkit: human computation algorithms on mechanical turk. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 57–66. ACM, 2010.

[6] Wendy E Mackay, Caroline Appert, Michel Beaudouin-Lafon, Olivier Chapuis, Yangzhou Du, Jean-Daniel Fekete, and Yves Guiard. Touchstone: exploratory design of experiments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1425–1434. ACM, 2007.