Using Feature Vector Representations To Identify Similar Projects In App Inventor

Maja Svanberg
Computer Science Department, Wellesley College
Wellesley, Massachusetts, USA
msvanber@wellesley.edu

***Abstract*—In trying to understand the big picture of how users learn to program in App Inventor, we want to be able to represent projects in a way suitable for large scale learning analytics. Here I present different representations of projects that could potentially be used to identify App Inventor projects that have structural similarities to each other, e.g., projects created by users following tutorials. I compare the different representations based solely on how accurately they predict the correct tutorial from a labeled data set. The results suggest that we use both blocks and components from a project, apply TF-IDF to the counts of each feature, and measure distance or similarity in terms of a generalized Jaccard distance. This work lays the foundation for being able to find clusters of similar projects to distinguish original from unoriginal projects and to be able to filter out similar projects when trying to determine a user's skill level.**

I. Introduction

App Inventor is an online programming environment that lets users build their own smart-phone apps using a blocks programming language. Formalizing a notion of structural similarity between projects enables us to apply large scale learning analytics to the App Inventor environment. This notion facilitates filtering out unoriginal [1] projects, e.g., projects created following tutorials, when analyzing projects for computational thinking and promises to be more effective than attempts (e.g., [2]) based solely on project names. A formal definition of similarity allows performing unsupervised machine learning algorithms and discovering linked projects, e.g., collocated classroom activities and collaborative projects between users [3]. Knowing which projects are unoriginal, we may be able to filter out these to better assess a user's skill level for particular constructs and concepts, e.g., procedural abstraction [4] Accurate identification of tutorials would make it possible to revisit work on skill progression in App Inventor ([5], [6]) to determine the impact of tutorials.

II. Data set

I use a data set consisting of 894 projects from 16 students who took a Fall 2015 Wellesley CS0 course based on App Inventor. Students created projects in lecture and in lab, as well as in five assignments that involved creating original projects. Their projects were compared to a set of 169 tutorials used in the course, drawn from the App Inventor website, App Inventor Maker Cards, and in-class exercises from the Wellesley course. They were labeled by category using name, date, and structural observations. 310 of the projects could not be identified as any specific tutorial, and are ignored in this study.

III. Method

I set out to explore how different representations of projects behaved when trying to establish similarity in relation to other projects. I look at similarity measures that differ by feature selections, normalization, and distance metrics between vectors. I evaluate these on the data to determine which is best, both for recall, and for having the potential to determine where to establish a threshold for what should be considered “the same“ project.

In App Inventor, users build projects using components, which define the user interface and functionality, and blocks, which define the behavior of the components. The feature selections consider which combination of blocks and components to use, both, just components, or just blocks. While my explorations included bigrams, consisting of a tuple of each block and its top block, here I consider only unigrams. For feature normalization, I consider TF-IDF (term frequency over inverse document frequency) feature scaling [7][8], binary values, and counts. TF-IDF is meant to give more weight to less common blocks, whereas binary values only indicates whether a block is present in the project. Where p and q are vectors, and D are the dimensions, the four distance metrics I am using, Euclidean, Cityblock, a generalized Jaccard (hereby referred to as Jaccard*), and Cosine, are defined as:

$$Euclidean(p,q) = \sqrt{\sum_{i\in D}(q_i - p_i)^2}$$

$$Cityblock(p,q) = \sum_{i\in D} \mid q_i - p_i \mid$$

,

$$Jaccard^*(p,q) = 1 - \frac{\sum_{i\in D} min(q_i,p_i)}{\sum_{i\in D} max(q_i,p_i)}$$

$$Cosine(p,q) = 1 - \frac{\sum_{i\in D}(q_i * p_i)}{\sqrt{\sum_{i\in D} q_i^2}\sqrt{\sum_{i\in D} p_i^2}}$$

I measure the results in terms of recall of how many projects were correctly classified as the right category based on a nearest neighbor approach.

IV. Results

The differences in how the difference metrics performed were small, but for our purposes, we can see in Figure 1 that out of the metrics, Cosine and Jaccard* slightly outperformed Euclidean and Cityblock. From Figure 2, it becomes clear

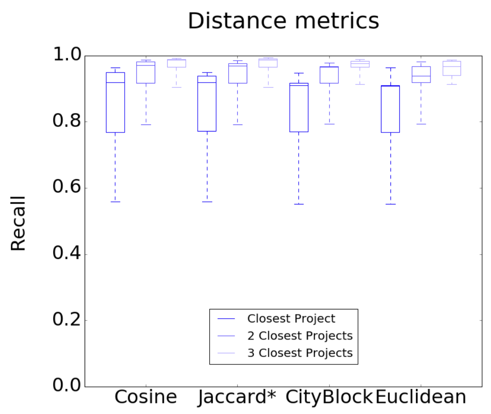

Fig. 1. Box plots of distance to the closest tutorial, grouped by the four different metrics. The three subsections per metric represents closest, 2nd closest, and 3rd closest tutorial. The whiskers include all projects.

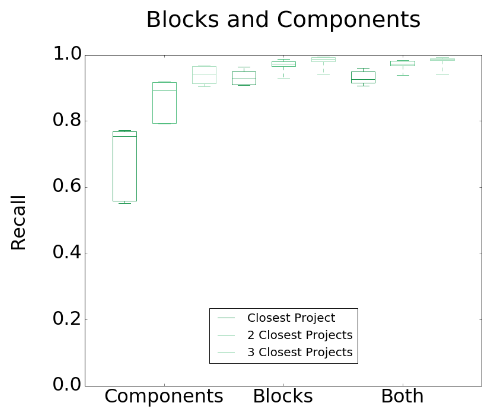

Fig. 2. Box plots of distance to the closest tutorial, grouped by the three combinations of blocks and components. See Fig.1 for details.

that components on their own are not a good measurement, and similarly adding them to representations that already accounts for blocks does not do very much. As for feature normalization, we can see in Figure 3 TF-IDF had the overall strongest performance.

V. Discussion and Future Work

Not only did they perform better than their counterparts, but Cosine and Jaccard* might be the most useful to find a normalized notion of similarity as they normalize all distances

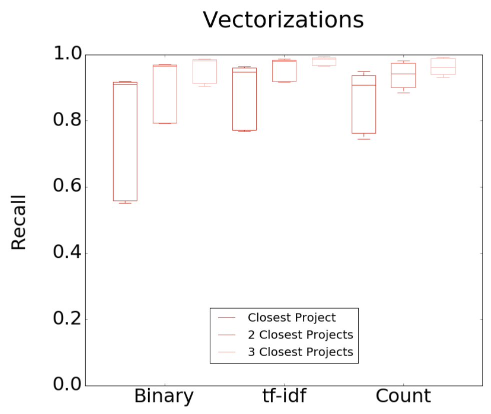

Fig. 3. Box plots of distance to the closest tutorial, grouped by the three normalizations. See Fig.1 for details.

to a value between 1 and 0. To distinguish between the two, we know that Jaccard* also takes into account difference in size, and hence, we might prefer Jaccard* to Cosine. Since components added very little to the recall, we can conclude that most of the characteristics of a project is in the blocks rather than the components. However, we might still want to represent the components in our vector, as they might distinguish the design of the screen, such as arrangements.

However, there are limitations:

1) The data set is small, and not a random subset of users. It is yet to be determined whether the patterns of those learning to use App Inventor in this classroom differ from those learning it elsewhere.
2) The tutorials I am using as references are not equidistant from each other.
3) Projects were manually labeled using a combination of creation date, project name, and structure. However, this might not necessarily reflect the user's intent of trying to follow a tutorial.

Going forward, I would like to find a "threshold" that would serve as a cutoff for finding similar projects "in the wild", be they tutorials, or other unknown patterns users follow. By finding these tutorials, we could automatically expand our labeled data set of tutorials, which would empower us to apply machine learning algorithms to gain more insight into user habits.

Acknowledgments

Thanks to Profs. Franklyn Turbak and Eni Mustafaraj for their continued support and advice. This work was supported by Wellesley College Faculty Grants and by the National Science Foundation under grant DUE-1226216. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author, and do not necessarily reflect the view of the National Science Foundation.

References

[1] E. Mustafaraj, F. Turbak, and M. Svanberg, "Identifying original projects in App Inventor," in *Proceedings of the 30th International FLAIRS Conference*, May 2017.
[2] B. Xie, I. Shabir, and H. Abelson, "Measuring the usability and capability of App Inventor to create mobile applications," in *3rd International Workshop on Programming for Mobile and Touch*, 2015, pp. 1–8.
[3] F. Turbak, E. Mustafaraj, M. Svanberg, and M. Dawson, "Work in progress: Identifying and analyzing original projects in an open-ended blocks programming environment," in *Proceedings of the The 23rd International DMS Conference on Visual Languages and Sentient Systems (DMSVLSS 2017)*.
[4] I. Li, F. Turbak, and E. Mustafaraj, "Calls of the wild: Exploring procedural abstraction in App Inventor," in *Proceedings of the IEEE 2017 Blocks and Beyond Workshop*, 2017, (to appear).
[5] B. Xie, "Progression of computational thinking skills demonstrated by app inventor users," MIT EECS Master of Engineering thesis, May, 2016.
[6] B. Xie and H. Abelson, "Skill progression in MIT App Inventor," in *IEEE Symposium on Visual Languages and Human-Centric Computing*, 2016, pp. 213–217.
[7] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.
[8] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.