Modeling Programming Knowledge for Mentoring at Scale

Anvisha H. Pai
MIT CSAIL
anvishap@mit.edu

Philip J. Guo
MIT CSAIL / University of Rochester
pg@cs.rochester.edu

Robert C. Miller
MIT CSAIL
rcm@mit.edu

ABSTRACT
In large programming classes, MOOCs or online communities, it is challenging to find peers and mentors to help with learning specific programming concepts. In this paper we present first steps towards an automated, scalable system for matching learners with Python programmers who have expertise in different areas. The learner matching system builds a knowledge model for each programmer by analyzing their authored code and extracting features that capture domain knowledge and style. We demonstrate the feasibility of a simple model that counts the references to modules from the standard library and Python Package Index in a programmers' code. We also show that programmers exhibit self-selection using which we can extract the modules a programmer is best at, even though we may not have all of their code. In our future work we aim to extend the model to encapsulate more features, and apply it for skill matching in a programming class as well as personalizing answers on StackOverflow.

Author Keywords
Skill matching; personalized learning; learner modeling

ACM Classification Keywords
H.5.3. Group and Organization Interfaces: Theory and Models

INTRODUCTION
Learning programming is an ongoing process. Knowing the syntax and built-in functions of a language is just the beginning. Even the most experienced programmers need to keep learning about new APIs, standard and external libraries, and frameworks that they can use in their programs. Instructional textbooks focus heavily on syntax and a few standard libraries, and can only go so far in helping a programmer learn something new. Moreover, it is easy for a programmer to get entrenched in certain practices, while there are better ones out there that they are never exposed to.

Interacting with other programmers who have different areas of expertise is a good way to address this problem [1]. Practices like pair programming, tutoring and mentoring are key. In structured classes, group projects reinforce this learning technique, as students have the opportunity to learn from each others' styles and expertise.

L@S'14, March 4–5, 2014, Atlanta, Georgia, USA.
ACM 978-1-4503-2669-8/14/03.
http://dx.doi.org/10.1145/2556325.2567871

However, finding peers or mentor guidance is not as easy or feasible at scale. Scale could be anything ranging from a large programming class at a university (250–1,000 students) to a whole community of programmers on the Internet. While it is easy to find tutors online that can help with broad areas like a CS1-style Intro. to Python, it is harder to find someone to help with specific application needs, such as using linear algebra functions of the Python library NumPy. Though learners have online help forums like StackOverflow [1] as a resource, forum answers are not personalized to the asker's experience level and often require non-trivial amounts of prerequisite knowledge.

In this work-in-progress, we present first steps toward a scalable learner matching system for Python based on programming knowledge and styles. Our system will analyze the corpus of code a programmer has written and build a fine-grained *knowledge model* of their expertise in different areas. This will enable it to match learners with mentors or with fellow students for collaborative learning projects in either a MOOC or large residential courses. Since code analysis is automated, this system is easily scalable, and has the potential to improve the learning experience for many programmers.

MATCHING BASED ON PYTHON MODULE USAGE
Python has a standard library of 226 modules that provide standardized functions like regular expressions, JSON file parsing, and email processing. Beyond the standard library, widely used and approved modules are included in the Python Package Index. These external modules include 24,147 libraries like Django, a major web framework for Python, and NumPy, a computational and scientific package. Assessing how well-versed a programmer is in using a particular module is one proxy measure of their expertise in that application area.

Data Collection and Analysis
As an initial investigation, we solicited Python code from 10 programmers. Three of these students were beginners who submitted code from their introductory Python class at MIT. Three were undergraduates who described themselves as intermediate to advanced level programmers, and submitted projects they had done outside of classes. The remaining 4 programmers were researchers at the MIT Computer Science and Artificial Intelligence Lab, and submitted code from their research. We used regular expressions to mine the code and count the number of function calls made from each module in the standard library and Python Package Index. We also asked the participants to rank their perceived expertise in the most popular packages that came out of this analysis, using an

Subject	1st Package		2nd Package		3rd Package	
	module	calls	module	calls	module	calls
Beginner 1	random	19	UserDict	8	string	7
Beginner 2	random	6	string	5	graph	4
Beginner 3	random	6	Queue	1	string	1
Undergrad 1	Django	95	httplib2	71	socket	62
Undergrad 2	os	12	copy	4	profile	3
Undergrad 3	re	31	mechanize	23	lxml	12
Researcher 1	os	20	json	13	re	10
Researcher 2	Django	158	relay	20	teleport	16
Researcher 3	util	52	config	25	json	9
Researcher 4	Django	187	os	20	re	16

Table 1. Top 3 packages for participants, ranked by number of function calls in their submitted code.

Subject	Top 3 Packages	
	Self-Reported	Ranking Calculated by Model
Beginner 1	random, string, os	random, string, numpy
Beginner 2	random, string, numpy	random, string, numpy
Beginner 3	numpy, string, random	random, string
Undergrad 1	string, django, random	django, numpy, random
Undergrad 2	numpy, random, os	os, random, numpy
Undergrad 3	numpy, random, os	os, random, numpy
Researcher 1	os, json, numpy	os, json, random
Researcher 2	string, django, os	django, os, json
Researcher 3	json, string, os	json, os
Researcher 4	numpy, unittest, random	django, os, random

Table 2. Perceived and actual rankings of modules by subject

online survey where they assigned a rank to each of 7 packages. We used these reported ranks to assess the validity of the estimated expertise from the analyzing submitted code.

Results
Table 1 shows the top 3 Python packages per participant ranked by the number of function calls in their submitted code, showing a wide variety of modules across different experience levels and areas of expertise. As expected, different applications and types of projects showed different modules being used. The beginner programming students mostly used the same packages, since their projects were in the context of similar class assignments and applications. This was very different for the more advanced programmers, however. Even though Undergrad 1 and Researcher 2 informally characterized themselves as Django-based web programmers, they used very different modules in their code, showing specific expertise and strengthening the idea of learning from peer guidance.

We did not have access to all the code the participants had written, hence we risk having an incomplete model of their expertise. However, the results in Table 2 indicate that the top 3 packages reported by the participants were mostly consistent with the top 3 packages from the code that they submitted. This implies that we can identify the strongest areas of expertise from the code submitted, and enables us to use this model for the application of mentoring, where the strongest skills of each potential mentor are important.

Self-selection could be a reason for this correlation between the top reported and calculated packages. In the study we asked people to submit code to us, and it is possible that they self-selected recent projects that they thought were good examples. This skew in might actually benefit a skill-matching system, since if a person wants to most accurately represent their skills for offering their services as a mentor, they would likely submit recent and accurate code. In contrast, analyzing *all* of their code might yield worse results, since older code might not be representative of current skills or interests.

APPLICATIONS AND ONGOING WORK
The simple model is that it does not normalize for comparison: using the same functions repeatedly may not signify expertise. We plan to eliminate this limitation by normalizing within a package and across different programmers, by gathering data through larger versions of the study presented here.We also plan to include more features that capture expertise and style, such as what percentage of functions in a package were used (coverage) and the use of object-oriented or functional programming.

Matching with Mentors
As seen in the results, even with a simple model that analyzes modules used, we can capture *to some degree* which packages a person is best at. A straightforward way to use this model for matching beginners with mentors is to simply match them with the people that are best at the packages that they want to learn.

Matching with Peers
Matching with peers is slightly more complex than with mentors because the aim is to pair together people with *complementary* knowledge and styles. Programmers can be matched by modeling each person as a vector with the dimensions as different modules and the magnitude as the normalized score from that module. Programmers could be matched based on the relative magnitudes and angles between different vectors. We aim to deploy our peer matching system in an MIT programming class based on group programming projects to validate our hypothesis.

Matching with Online Resources
An application of the model we plan to develop would be StackOverflow customization. Especially for beginner programmers, it can be problematic when the top voted answer to a question on StackOverflow uses libraries they do not understand. With a model of the programmer's knowledge, we can assess which answer falls closest to what they are familiar with. StackOverflow answers have code snippets, which make it possible to analyze these snippets in the same way code is analyzed, reducing answer customization to a similar matching problem

ACKNOWLEDGMENTS
This work is supported in by the MIT SuperUROP program, funded by a donation from Robert Fano; and by Quanta Computer as a part of Philip Guo's postdoc funding.

REFERENCES
1. Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G., and Hartmann, B. Design Lessons from the Fastest Q&A Site in the West. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 2857–2866.