2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement

On the Use of Input/Output Queries for Code Search

Kathryn T. Stolee
Department of Computer Science
Iowa State University
kstolee@iastate.edu

Sebastian Elbaum
Department of Computer Science and Engineering
University of Nebraska–Lincoln
elbaum@cse.unl.edu

***Abstract*—*Context:* Programmers frequently compose keyword queries as they use information search engines to look for source code. This syntactic approach to code search is often imprecise and results in wasted efforts looking through irrelevant results. Semantic code search approaches aim to address this weakness by formulating queries that specify behavior, rather than keywords. A recent approach uses input/output examples as queries that illustrate the behavior of desired code. The technical feasibility of this approach has been illustrated, yet the impact of the change in the query model has not been assessed.**

***Objective:* We explore the cost and accuracy of using input/output queries for code search from the perspective of the programmer, considering two programming languages, Yahoo! Pipes and SQL.**

***Method:* We perform a controlled user study with 109 participants from two groups, students and Mechanical Turk, to assess the cost and accuracy of using input/output search queries.**

***Results:* Our results show that programmers can compose input/output queries in the targeted domains with over 92% average accuracy and in less than two minutes.**

***Conclusion:* The use of input/output queries does not seem to limit the early promise of semantic searches that depend on it.**

I. Introduction

Consider a programmer who wants to extract salary information for employees from a database using SQL. The programmer has two tables, one called `employee` with fields `[id, name, address]`, and another called `payroll` with fields `[id, account, salary]`. The goal - extracting salary information - is quite syntactically different from the solution - using an implicit join on the two tables based on `id`. Without knowledge of the SQL language and relevant keywords (e.g., *join, on*), finding a solution with a keyword query is difficult.

Instead of a keyword query, a semantic approach to code search requires a behavioral query. The most recent approaches require example input(s) and output [1][2]. Using the previous SQL example, the programmer could provide tables from a working database as input and select the data that should exist in the output (i.e., a table with fields `[id, name, salary]`).

Using input/output queries for code search has been shown to be technically feasible in visual programming languages such as Yahoo! Pipes, query languages like SQL, and object-oriented languages like Java [2]. This previous work, however, does not assess the impact of the change in query model. We do not know whether these semantic queries can be composed efficiently and accurately by programmers, which is critical to the success of this code search approach in practice.

In this work we begin to evaluate the viability of input/output queries in two programming languages, Yahoo! Pipes and SQL, where searching for source code with keywords can be challenging. *Our key contribution is an assessment of the cost and accuracy of formulating output from an input in Yahoo! Pipes and SQL through a controlled quasi-experiment.*

II. Background

We introduce the two domain-specific, end-user programming languages used in this study, Yahoo! Pipes and SQL.

A. Yahoo! Pipes

Yahoo! Pipes is a mashup language with over 90,000 users [3] and a public repository of over 100,000 artifacts [4]. These programs combine, filter, sort, annotate, and manipulate RSS feeds. Programmers write Pipes programs in the Pipes Editor, dragging and dropping predefined modules and connecting them with wires to define the data and control flow. An example pipe is shown in Figure 1. The *Fetch Feed* modules provide lists of RSS feed items to the pipe. Each item is a map data structure with key-value pairs. An example RSS feed with only three items shown is in Figure 3. Each pipe can have multiple source modules that access data sources (e.g., URLs), and one sink, the *Pipe Output*. In Figure 1, the pipe concatenates two data sources with a *Union* module and retains items that contain the word "tennis" using a *Filter* module.

1) Existing Search Capabilities: In the state-of-the-practice, programmers can only search for pipes within the Pipes environment by URLs accessed, tags, keyword, or modules. Searches can return thousands of results, which is not surprising as many mashups access common feeds or websites. Searching through generic web search engines is not viable as the pipes reside in a repository that has a proprietary format.

2) Input/Output Query Model: Our previous work has suggested that input/output can be used as a query model when searching a repository for existing Yahoo! Pipes programs [1]. In this domain, the inputs are URL(s) from which the approach gathers RSS feeds forming input list(s), and the output is a combined and modified list of items from the input list(s).

B. SQL

SQL select statements have been used for decades to support data retrieval, operating on their own or being embedded into applications. Given the simplicity of the SQL syntax and its popularity, even well conceived syntactic searches for examples will return many irrelevant results.

1) Existing Search Capabilities: Programmers can find many resources related to SQL using information search engines. In a survey of 109 programmers (administered as part of the study that follows), 20 participants reported to use SQL frequently. Those programmers all perform programming tasks at least weekly and 70% search for code at least weekly.

978-0-7695-5056-5/13 $26.00 © 2013 IEEE
DOI 10.1109/ESEM.2013.35

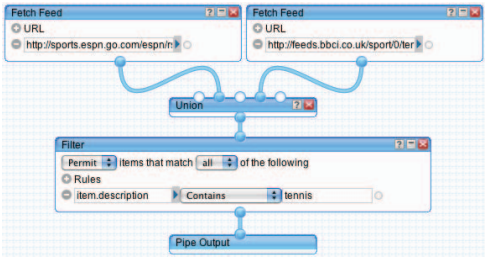

Fig. 1. Example Yahoo! Pipes Program

2) Input/Output Query Model: When instantiating the semantic search for SQL, our previous work has suggested that the input and output are database table(s) [2]. The output table would be selected column(s) and row(s) from the input table(s); an example input table is shown in Figure 3. An example select statement to retrieve the first three rows as output could be:

```
SELECT * FROM table WHERE id <= 3;
```

III. Study

In this section, we describe a controlled user study to assess the use of input/output as a query format in Yahoo! Pipes and SQL, toward the following research questions:

RQ1: How *accurately* can programmers compose input/output specifications?

RQ2: How *efficiently* can programmers compose input/output specifications?

A. Design

We designed an experiment with ten tasks. Each task presents an input and a problem description and asks the participant to select the expected output using checkboxes. Five tasks were created for each domain. The posttest measures were accuracy and time.

Figure 2 shows a task for Yahoo! Pipes. The problem description is, "Select the third-most-recent item from the list," and the input list consists of five items. The expected output is the third item. Figure 3 shows a task in SQL. With the description, "Select the rows with a price per unit greater than $1.00." Rows 3, 5, 6, and 8 form the expected response.

Note that the participants only specify the output, yet we aim to evaluate the input/output specification model as a whole. The rationale for this choice is twofold. First, for the domains being evaluated, the inputs can easily be obtained by the programmer from their context (i.e., a URL for Yahoo! Pipes, or a database table for SQL). Starting with an existing input with the challenge of computing an output mimics how a semantic search could work in practice for these domains. Second, by fixing the input we can automate the accuracy assessment since we know the expected output. This reduces the cost of executing the experiment and the chances of accuracy measurement errors. We discuss this threat to construct validity in more detail later.

B. Task Creation

Each task requires three objects: the problem description, input, and expected output. To create realistic tasks, we begin with programs (artifacts) from which these objects are created.

We collected five Pipes programs intended to be "typical" based on structural uniqueness and popularity, and reused from

Select *the the third most-recent item* from the list

Title:	As space junk falls, Russia hints at sabotage
Description:	A space probe stuck in orbit could fall back to Earth as soon as Sunday or Monday, though most experts say the chance that debris will harm anyone on the surface is slim. And even as the Russian space agency released new forecasts of the probe's...
Link:	link
Date:	Fri Jan 13 11:49:00 CST 2012
Title:	Warm today; red-flag warning for mountains
Description:	Warm, dry winds will raise wildfire risk in the Santa Ana Mountains Friday, prompting a red-flag warning from the National Weather Service. Gusts from the northeast as high as 45 to 50 mph could rake the windiest spots in canyons, passes and slopes...
Link:	link
Date:	Fri Jan 13 08:49:00 CST 2012
Title:	Red-flag warning for Santa Ana Mountains
Description:	The National Weather Service has issued a red-flag wildfire warning for the Santa Ana Mountains from midnight Thursday to 2 p.m. Friday.Â¬â€ prompted by expectations of gusty winds and low humidity. High-pressure air over the Great Basin is ratcheting...

Fig. 2. Task 4 in Yahoo! Pipes

Select **the rows with a price per unit greater than $1.00** from the table below, using the checkboxes next to each row.

	Id	Fruit	Variety	Vendor	Price	Unit
☐	1	Apple	Fuji	John's Produce	$0.67	lb
☐	2	Apple	Jonathan	John's Produce	$0.84	lb
☐	3	Apple	Fuji	Fran's Fruit	$1.23	kg
☐	4	Pear	Green Anjou	John's Produce	$0.98	lb
☐	5	Pear	Bartlett	Pearly Pears	$2.01	kg
☐	6	Grapes	Champagne	Eduardo's Uvas	$5.21	kg
☐	7	Bananas	Plantain	Pato's Plantains	$0.50	lb
☐	8	Melon	Honeydew	Oregon Melon, Inc.	$1.10	lb

Fig. 3. Task 6 in SQL

a previous study [1]. The input was derived by extracting the URL(s) and accessing the RSS feed(s). Executing the pipe generates the output list, which is used as the oracle. The problem description was generated by the researchers to reflect the pipe semantics. Table I shows the problem descriptions for Yahoo! Pipes Tasks 1-5, along with the input and output sizes. To capture the behavior of the pipes while keeping the input size reasonable, we limit the number of items from each RSS feed to seven. String lengths were bound to 100 characters.

SQL artifacts were created since there is no existing reference repository. Artifact creation was based on language coverage, using common constructs in SQL select statements, including inequality and conjunction in the $WHERE$ clause, the $GROUP\ BY$ clause, and the $count$, $distinct$, and avg functions. The input and output are generated so that the output is returned when the statement is executed on the input; the output forms the oracle. Problem descriptions were generated by the researchers to reflect the semantics of the statement, as shown in Table I for SQL Tasks 6-10. The input table is the same for all SQL tasks, but the input sizes differ. Task 6 asks the participant to select rows, so the input size is 8. Task 7 asks about columns and rows, forming 14 potential selections.

C. Participants

The participants in this study were solicited from two populations, undergraduate computer science classes at UNL and workers on Amazon's Mechanical Turk [5]. Of the 109 participants, 43 came from junior/senior undergraduate classes at UNL and 66 came from Mechanical Turk.

D. Implementation

This study delivery was based on the participant group. Students performed the study in the classroom. Mechanical Turk workers performed the study online. The first part of the study involved a 10-question survey about programming and search habits. The Mechanical Turk workers were paid

TABLE I. EXPERIMENTAL TASKS DESCRIPTIONS AND RESULTS

Task	Domain	Problem Description	Input Size	Output Size	Accuracy n	Accuracy Mean	Accuracy Median	Timing (m:ss) n_2	Timing Mean	Timing Median
1	Pipes	Select all records that show the Current Weather Conditions or the 10-Day Forecast for Malibu, Exeter, or Camarillo	21	6	65	90%	94%	30	2:30	1:48
2	Pipes	Select the four most-recent records that contain information about a hotel	21	4	63	87%	90%	24	3:48	2:55
3	Pipes	Select the first three records from each source, where the sources are indicated using different background colors	14	9	65	93%	100%	29	1:20	0:46
4	Pipes	Select the the third most-recent record from the list	5	1	72	96%	100%	30	1:14	0:47
5	Pipes	Select all records with the pink background, and those items from the grey background with "au" in the description	11	4	70	95%	100%	30	2:26	2:05
6	SQL	Select the rows with a price per unit greater than $1.00 from the table.	8	4	72	99%	100%	30	0:41	0:34
7	SQL	Select the id, fruit, and variety for each item priced between $0.75 and $1.25 per unit from the table.	14	7	71	88%	93%	30	1:37	1:29
8	SQL	Count the number items that are priced by kg (instead of lb) from the table.	5	1	63	86%	100%	21	1:08	0:52
9	SQL	Compute the average price for each fruit from the table.	5	1	68	94%	100%	26	1:44	1:21
10	SQL	Identify which vendors sell apples from the table.	5	1	72	92%	100%	30	1:22	1:03

whereas the student participants were not. Participation was restricted to those who accepted the informed consent.[1] Not every participant completed every available task.

1) Classroom: The classroom implementation required paper packets that delivered the IRB informed consent, survey, and experimental tasks. Each packet was numbered at random. Students had 15 minutes in total to complete as much of the packet as possible. The first pages of the packets contained the informed consent and survey. The experimental tasks followed and the order was the same for all packets, starting with the SQL Tasks 6 - 10, and then Yahoo! Pipes Tasks 1 - 5.

2) Mechanical Turk: The survey was implemented as a qualification exam for Mechanical Turk participants. There were additionally two competency questions about Yahoo! Pipes and SQL that had to be answered with 50% accuracy. The IRB form was signed electronically. A passing score and IRB acceptance allowed the worker access to the experimental tasks. Each experimental task in the study was implemented as a *human intelligence task*, or HIT. Participants had a maximum of 10 minutes to complete each HIT, and were paid $0.26 if their work was accepted. To prevent participants from 'gaming' the tasks, 50% accuracy was required for payment. The study was available for three weeks.

E. Metrics

Time and accuracy were collected per task. Time was collected by the Mechanical Turk server; we did not collect per-task timing for classroom participants. Accuracy was measured by scoring the participant responses against the oracles. One point was awarded for each input correctly selected or not selected to be part of the output. The score awarded was a percentage out of the total points (i.e., the input size). For example, for an input with five items, if the oracle has item 3 selected and the participant selects items 2 and 3, the score is 4/5 = 80%. In Mechanical Turk, the participant would have gotten paid for this work since the score is at least 50%.

IV. RESULTS

A. Yahoo! Pipes

Accuracy and time for Tasks 1 - 5 are summarized in Table I. The total number of participants per task is in the *n* column, followed by the mean and median accuracy aggregated across all participants (*RQ1*). An average of 67 participants performed each Pipes task, with a range from 63 to 72. The average accuracy was 92%; Task 4 had the highest average accuracy at 96%. Task 4 also had the smallest input and the smallest output; it turns out there is a negative correlation between accuracy and the input size (Spearman's $r = -0.2946$), but not between the accuracy and the output size ($r = -0.0136$).

For *RQ2*, n_2 (the number of Mechanical Turk participants) ranged from 24 to 30. The average time per task was 2:12, with a range from 1:14 to 3:48. The timing appears to be dependent, in part, on the size of the input. The Spearman correlation between the time and input size reveals a strong relationship $r = 0.3565$ (correlation for time and output was $r = -0.0397$).

To give more context to the timing data of formulating an input/output query, consider that understanding pipes of similar complexity may take on the order of 16 minutes [6]. Although full understanding of a pipe is not needed to discard irrelevant matches, the cost of this pruning activity is such that investing in a query that takes a couple of minutes but only returns semantically relevant results seems promising. A comprehensive study on the input/output query against other query models will be one of the targets of our future work.

B. SQL

Accuracy and timing information for Tasks 6 - 10 are summarized in Table I. An average of 69 participants completed each of the SQL tasks. Across all SQL tasks, as with Yahoo! Pipes, the average accuracy was 92% (*RQ1*). SQL participants performed best on Task 6, with an average score of 99%. The lowest accuracy came from Task 8, with an average of 86% and a median of 100%. Since the input table was the same for all the SQL tasks, we could not draw conclusions about the relationship between accuracy and input size.

For *RQ2*, timing data reveals an average completion time of 1:18. Participants provided the fastest responses on Task 6, with an average of 41 seconds and a median of 34 seconds. Task 9 took the longest, with an average of 1:44. In general, longer times were associated with lower accuracy (Spearman's $r = -0.1970$). It likely took the participants longer to guess the answer than if they knew the answer outright.

V. DISCUSSION

The focus of this work was on assessing the cost and accuracy of using input/output queries, and we discuss the implications and threats to validity in this section. In addition, some interesting results emerged pertaining to the use of two diverse populations, students and Mechanical Turk workers.

[1]The tasks, informed consent. and UNL Institutional Review Board approved process can be found at http://cse.unl.edu/~kstolee/iostudy/Study.html.

TABLE II. DIFFERENCES IN RESULTS BASED ON IMPLEMENTATION. $H_0 : \mu_{mt} = \mu_s$

Task	Mechanical Turk	μ_{mt}	Students	μ_s	p-value
YP	143	0.943	192	0.908	0.0095***
SQL	137	0.916	209	0.926	0.4160
Overall	280	0.929	401	0.917	0.2122

$\alpha = 0.1$* $\alpha = 0.05$** $\alpha = 0.01$***

A. Comparing Results

Since this study involves two different implementations, we examined any differences in accuracy based on the population.

1) Analysis: We segmented the results based on population and computed the mean accuracy for each domain and overall. The mean for Mechanical Turk is μ_{mt} and the mean for the students is μ_s. We performed a Mann-Whitney-Wilcoxon[2] test with the null hypothesis, $H_0 : \mu_{mt} = \mu_s$.

2) Results: Table II presents the results of this analysis. Overall, no difference between the groups was observed. This provides support for the use of large-scale and cost-effective crowdsourcing environments like Mechanical Turk to support empirical studies that serve at least to complement other studies.

Care needs to be taken, however, as the population gets smaller or the requirements more specific. Splitting the analysis by domain reveals some differences.

Yahoo! Pipes: We reject the null hypothesis that there is no difference between the populations at $\alpha = 0.01$. Student performance is lower than Mechanical Turk performance, and we conjecture this is a result of the delivery. For the students, we used paper packets of tasks, while for the Mechanical Turk participants we had online versions. For Task 2 and Task 3, both of which showed significant differences at $\alpha = 0.01$ and $\alpha = 0.05$, respectively, the list of input items spanned multiple pages on paper. On the web version, the participants could scroll and search for keywords with the browser find function. This may have contributed to a lower student performance.

SQL: We did not reject the null hypothesis. However, the average Mechanical Turk score was lower than the student score, even though μ_{mt} may be artificially high, given that we did not consider the results when the accuracy was less than 50% (25 HITs were rejected). Among the students, only three tasks were below 50%. So, there may be an unobserved significant difference between the participant groups.

B. Threats to Validity

1) Internal: The results are subject to self-selection bias. The participants chose which, and in what order, to complete the tasks. Additionally, Mechanical Turk workers were only paid if their accuracy was at least 50%. This was for quality control, but it also biases the accuracy measurements.

Instrumentation may have played a role in the results. It may have lead to increased accuracy since we are not over-constraining the study context, on the other hand, some tasks may be better suited for one context than another.

2) External: We have evaluated the input/output queries in two domain-specific, end-user programming languages, Yahoo! Pipes and SQL. The extent to which this query model extends to other programming languages is yet to be explored.

[2]We tested the normality of the data using the Shapiro-Wilk test with a null hypothesis that the population is normal. The null hypothesis was rejected at $\alpha = 0.05$, and so we used the non-parametric Mann-Whitney-Wilcoxon test.

We selected experimental tasks with the goal of representing "typical" tasks. However, particularly for the SQL domain, the tasks may not be representative of actual user tasks. Replication of this study with other tasks is needed to generalize further.

The participants in our study were Mechanical Turk workers and computer science undergraduate students. While only 8% of the participants had no programming experience, these populations may not be representative of programmers who search for code in the evaluated domains.

3) Conclusion: Accuracy and timing data were highly dependent on the quality and clarity of the problem description. Despite best efforts, these measures may be unreliable.

4) Construct: Participants only specified the output in the tasks. For the domains evaluated, the input can easily be obtained from the programmer's context, so this mimics how our approach could be used in practice. In other domains, this may not be the case so the format may not generalize.

Accuracy for Tasks 8, 9, and 10 may be inflated. We did not collect results for tasks if the accuracy was lower than 50% and omitted the results of 12, 8, and 5 participants, respectively.

VI. CONCLUSION

We have conducted what we understand to be the first assessment on using input/output queries for search, which are one of the key components of emerging semantic code search techniques, and also of other approaches used in program synthesis. The findings from our experiment with 109 participants performing up to ten experimental tasks reveal that such queries can be performed with over 90% accuracy and within a very reasonable time investment considering the potential gains. That said, there are several areas of the study that we would like to revisit and extend, as captured by the discussion of the threats to validity. In addition, we realize that code search is a complex process that is not just about query formulation or result analysis, but a combination of those activities under various contexts, tools, and programmer activities. Hence, we would like to increase the scope of our studies to incrementally incorporate such factors.

ACKNOWLEDGMENT

This work is supported in part by NSF Award SHF-1218265, NSF Graduate Research Fellowship under CFDA-47.076, and AFOSR Award #9550-10-1-0406.

REFERENCES

[1] K. T. Stolee and S. Elbaum, "Toward semantic search via smt solver," in *Symposium on the Foundations of Software Engineering*, 2012, pp. 25:1–25:4.

[2] K. T. Stolee, S. Elbaum, and D. Dobos, "Solving Semantic Searches for Source Code," University of Nebraska-Lincoln, Tech. Rep. TR-UNL-CSE-2012-0012, November 2012.

[3] M. C. Jones and E. F. Churchill, "Conversations in Developer Communities: A Preliminary Analysis of the Yahoo! Pipes Community," in *International Conference on Communities and Technologies*, 2009.

[4] "Yahoo! Pipes," http://pipes.yahoo.com/, June 2012.

[5] "Amazon Mechanical Turk," https://www.mturk.com/mturk/welcome, June 2010. [Online]. Available: https://www.mturk.com/mturk/welcome

[6] K. T. Stolee, "Analysis and Transformation of Pipe-like Web Mashups for End User Programmers," Master's Thesis, University of Nebraska–Lincoln, June 2010.