Connecting Stack Overflow code snippets with API documentation

David Xu
University of Waterloo
DC 3594
Waterloo, Ontario N2L 3G1
d44xu@uwaterloo.ca

Adriaan Labuschagne
University of Waterloo
DC 3334
Waterloo, Ontario N2L 3G1
alabusch@uwaterloo.ca

ABSTRACT

Stack Overflow contains thousands of questions and answers about Java programming APIs. The answers often contain code snippets that can be considered API usage examples.

Unfortunately, there is no link between these usage examples and API documentation.

In this paper we make three contributions (1) we augment Stack Overflow answers by providing documentation for the APIs used in their code snippets (2) we provide usage examples of Java types by injecting links to Stack Overflow questions into the Android docs and (3) we identify related Stack Overflow questions based on the methods and types used in their snippets.

We show how our contributions can reduce the time required to find API documentation, can add useful API usage examples to documentation, and can improve upon Stack Overflow's current method of finding related questions.

Keywords

API Documentation, Java, Stack Overflow

1. INTRODUCTION

Contemporary software developers make extensive use of Application Programming Interfaces (APIs). The use of APIs is often necessary (e.g., consuming a web service) and allows developers to reuse existing libraries thereby reaping the benefits of others' coding and testing efforts. Using APIs, however, is not without cost as both novices and experts face obstacles while learning and coding to an API [5].

Developers face difficulties often stemming from a lack of documentation and API usage examples [6], and writing more documentation and keeping it up to date is a difficult task [8 and 9 in live api].

To address these issues we make the following contributions:

1. The addition of API documentation directly to Java snippets on Stack Overflow.
2. Identification of related Stack Overflow posts based on the APIs used in code snippets.
3. The provision of API usage examples to the official Android API documentation.

The key ingredient that makes our contributions possible is knowing what types and methods are used in code snippets found on Stack Overflow. Subramanian et al. [8] built a tool called Baker that analyzes code snippets of arbitrary length and finds the fully qualified names (FQNs) of code elements used therein. An example code snippet that Baker can parse is shown in Figure 1. Baker was able to identify the FQNs of `setKey` and `getKey` whose FQN, as an example, is `android.preference.Preference.getKey`. 89% [2] of unqualified Java method names and 32% [9] of type names are ambiguous, an example from Subramanian et al. is that the unqualified method name `Log` can refer to either `che.tomcat.util.log.Log` or `org.eclipse.jetty.util.log.Log`, among many others. Baker can often identify the exact match despite this ambiguity. 69% of the elements Baker identified from a random set of 4000 Stack Overflow snippets were identified exactly and a list of possible FQNs were found for the remaining 31% [9].

```
SharedData data = SharedData.getInstance();
data.setKey("some string");//to set value
data.getKey(); //to get value
```

Figure 1: Code Snippet S4

Subramanian et al. briefly explore the utility in using JavaScript to augment StackOverflow and Android documentation. Our first contribution provides a more streamlined linking to enhance the usability. Our third contribution explores using clustering on FQNs to find the most useful API usage examples. Contribution two is our own initiative.

For our first contribution, we use the FQNs that Baker finds in Android code snippets to fetch documentation from the official Android webpage in order to annotate Stack Overflow code snippets. Section 3 provides a scenario to describe

the usefulness of the documentation and details on how we implement this. In Section 4 we describe how we use clustering on the FQNs found in snippets to identify related Stack Overflow questions, while Section 5 describes how we link to Stack Overflow code snippets from the Android documentation. Section 2 provides details on the dataset we used. We evaluate our contributions in Section 6, and discuss our results in Section 8.

2. BACKGROUND

In order to run our clustering algorithms on the FQNs found in Stack Overflow (SO) posts we required a large database that associates FQNs with the posts they're found in. The researchers that created Baker provided us with a database that they built from the official SO data dump of 2012. All snippets found in the accepted answers of questions tagged `Android` were parsed by Baker. The returned FQNs as well as the line numbers they were found in were stored. 7,880 methods and 1,673 types were identified in 18,265 accepted answers. Only snippets in accepted answers were parsed as these are likely of higher quality and are therefore better examples of API usage.

3. DOCUMENTING CODE SNIPPETS

3.1 Scenario

An Android developer posted a question on Stack Overflow for which he received the answer shown in Figure 2. The answer contains a code snippet with several methods and types. Our developer faces a problem; he doesn't know the details behind the `postRotate` method. Usually he would search Google for the `Matrix` class and the `postRotate` method which would lead him to the Android documentation. This involves several steps, and still leaves it up to the developer to find the correct `postRotate` method since there are several overloaded methods with that name in the `Matrix` class.

This presents an opportunity to improve on the developer's experience. We can inject documentation into the page as shown in Figure 2. The popup displaying an accordion and documentation for the `postRotate` method appears when the developer clicks on the line of code in question. There is now no need to search the web for documentation and the developer can easily shift his attention between the documentation and the snippet without having to switch tabs or reorganize windows. As described in the introduction, Baker can often determine the exact FQN without the type (e.g., `Matrix`) being present in the snippet, which is a situation that can create confusion for the developer if the method name is ambiguous.

3.2 Method

To add documentation to a snippet we built a Google Chrome extension that allows us to run a JavaScript script every time a user visits Stack Overflow. The extension extracts the SO answer id which is embedded in an HTML tag from the page. It then makes an HTTPRequest to the snippet database described in the Background section to retrieve the FQNs found in the snippet and the line numbers they appear in. We now turn the FQNs into URLs, for instance `android.view.Window.findViewById(int)` becomes `http://developer.android.com/reference/android/view/Window.html#findViewById(int)`. This URL can now be used to fetch the page which can then be parsed to extract the method's documentation. Documentation is stored in a hash and displayed once a user clicks on a line of code. Although we currently display documentation for every FQN found in the line it is possible to improve the user experience by underlining specific methods and types which will allow the user to see documentation for the specific element he/she clicked on.

When there isn't enough context in a snippet, Baker may not be able to match a method or type with a single FQN and instead returns all possible matches (i.e., guesses). We decided to provide these guesses as it still helps to speed up the user's search process. A link to the documentation is also provided in case the user needs further details.

Baker exposes an API which would allow us to parse and therefore document any snippet found on Stack Overflow. We chose to query the snippet database instead of using the API because Baker often takes a minute or more to parse a snippet resulting in an unacceptable delay. Using our preparsed database we can document snippets almost instantly. This does not present a true limitation as Baker could be extended to store any Stack Overflow snippet's FQNs in a database the first time it is parsed. This allows subsequent requests to be served from this database, thereby avoiding the parsing delay.

4. SUGGESTING RELATED POSTS

4.1 Scenario

When looking at a question on Stack Overflow the user is provided with links to related questions in the page's sidebar. This can provide questions to help improve code understanding and even API discoverability for the user. Stack Overflow's related questions are based on a full text search on the database of questions [7] .This can be helpful when the context of the question is expressed well in the question and not in the code snippet. By leveraging Baker, we are able to perform code snippet comparisons based on the FQNs found in the code snippets instead. This provides an added dimension to the related post searches. The ideal scenario is to find related posts based on similar FQN calls in both questions. Our developer may benefit from related questions that are more oriented towards the APIs in the snippets, rather then the plain-text of the question. An example of the interface is shown in Figure 3.

4.2 Method

Relating Stack Overflow questions with each other can be done by simply comparing the FQNs in the code snippets with each other. This can provide a count between how many FQNs match between one question and another.

One issue with this naive approach is that FQNs are equally weighted for comparison. For instance, if one question contains a code snippet with `java.lang.String.valueOf(long)` it would have a high relation to another code snippet that contains the same method. But in fact, the questions can be completely different because `java.lang.String.valueOf(long)` is a very generic and commonly used method to convert a Java Long type to a String type. This is probably not related to the context of the question.

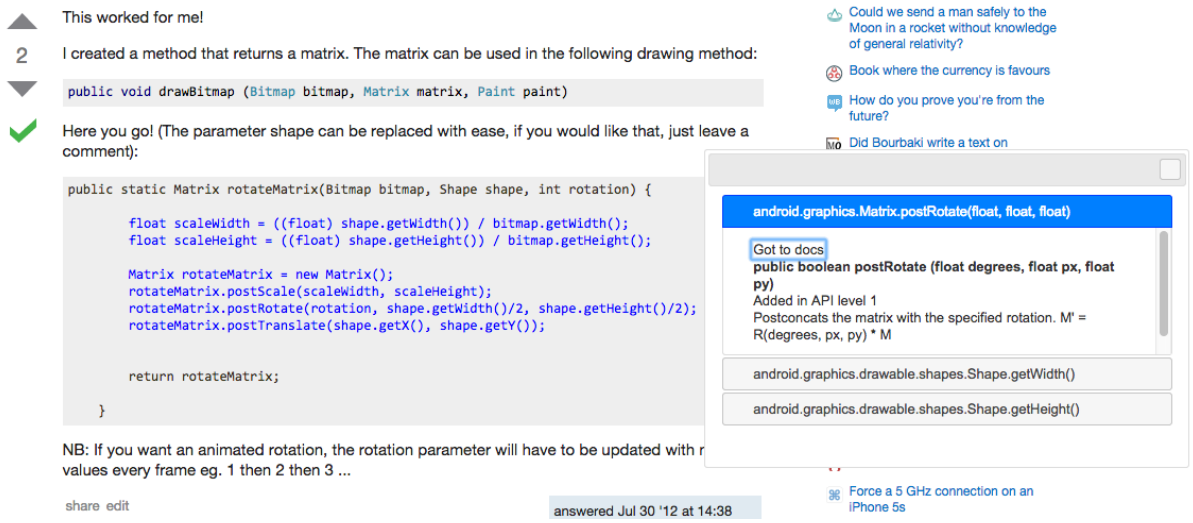

Figure 2: A screen shot of injected documentation on Stack Overflow

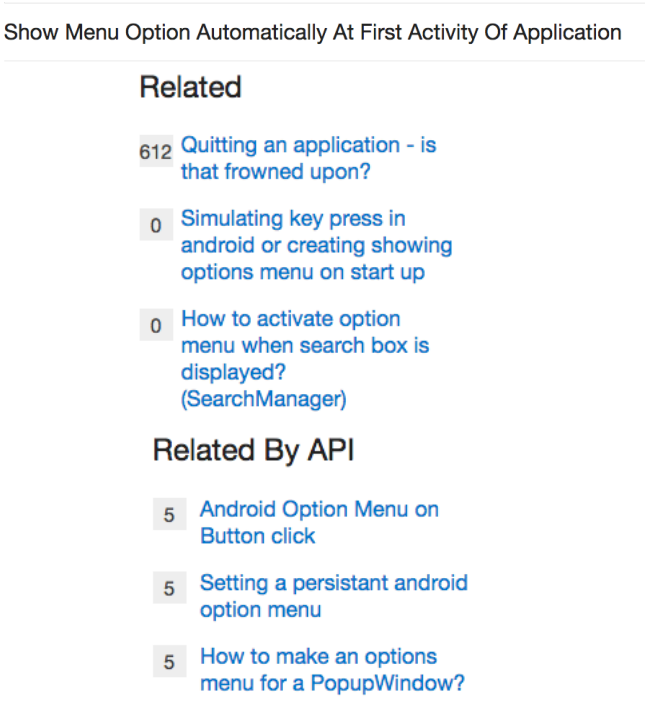

Figure 3: An augmented screen shot of a Stack Overflow post, showing the question title, the related questions that Stack Overflow supplies, and the related questions our tool suggested on the sidebar of the question. Our tool injects suggestions right under Stack Overflow's related questions.

On the other hand, a question with a code snippet containing `android.app.Activity.openOptionsMenu()` is more likely to be relevant with a code snippet containing `android.app.Activity.openOptionsMenu()`. This method is less common in code snippets. The intuition is that this uncommon method has more to do with the context of the question than a commonly used method would.

Table 1: Example idf scores for FQNs

Method FQN	idf score
java.lang.String.valueOf(long)	4.43
android.app.Activity.openOptionsMenu()	7.17
android.net.wifi.WifiManager.getDhcpInfo()	9.81

In order to capture this concept we implement the use of term-frequency inverse-document-frequency (tfidf) as a weighting for FQNs [4]. This is a common technique used to rate how important a word is in a document. We calculated a tfidf value for a given FQN as follows:

$$tfidf = f_{t,d} \times \log \frac{N}{n_t} \quad (1)$$

where the $f_{t,d}$ is the term frequency which represents the amount of times the FQN shows up in the document. And the second term represents inverse document frequency (idf), where N represents the total amount of Stack Overflow documents in our dataset, and n_t represents the number of Stack Overflow documents where the FQN appears in our dataset.

With this equation we are able to improve the naive approach by incorporating the importance of each FQN. When comparing one question with the set of questions, we compare the matching FQNs. Instead of a simple count between matching FQNs, we sum the tfidf scores of matching FQNs in order to account for the importance of each FQN. Table 1 shows the idf scores for a common and uncommon APIs. Uncommon APIs have a higher weight when calculating question relevancy.

Unfortunately due to the dataset available, we were only able to calculate the tfidf scoring of the snippets in the accepted answers of the questions. Our hope here is that the FQNs used in the answer's snippet reflects the question's topic. Regardless we are still able to provide related questions based on the answers. We have augmented the Stack Overflow section to show the top three related questions based on FQN relations.

5. SUGGESTING API EXAMPLES

5.1 Scenario

Understanding APIs can be facilitated by referring to the documentation. This is often considered an official and proper source for understanding how to use an API as documentation is often created by the authors of the API. One thing to consider is that documentation is often static and may sometimes lack proper examples on how to use the class or method. For instance, Android documentation often lacks examples of API usage. The android class android.location.LocationManager is used to access the GPS data on an android phone. The official android documentation on this class outlines all usable parameters and methods, but does not provide an example code snippet [1]. In this section we decide to focus on the Android type (class) instead of methods.

This data can be filled by leveraging the dataset from Baker, as the dataset provides the FQNs of types used in the code snippets. Our intuition is to provide links to these Stack Overflow answers containing the use of the FQN. This has previously been done [9], but we intend to improve on it by ranking the examples by view count, and how detailed the examples are, using what we call snippet complexity. Our developer now has an option to explore Stack Overflow snippets that contain the Android API in question, where the snippets are ranked by popularity or snippet complexity. The augmented Android documentation interface is shown in Figure 4.

5.2 Method

5.2.1 *By views*

We decided to rank links to Stack Overflow questions by the number of views the questions have received. Our intuition is that popular questions likely contain snippets that demonstrate functionality that confuses a large number of developers. It may therefore be beneficial for a user exploring a class to view such questions in order to avoid pitfalls that may be inherent to design flaws in the API. Links to the top questions are listed in a table that a Google Chrome extension injects into the Android documentation.

5.2.2 *By complexity*

We applied the same concept of tfidf weighting described in the previous section. Each Android FQN was iterated through to calculate the idf values. For each Android FQN, we found answer snippets in the dataset that contained the FQN and summed the tfidf weightings of every FQN in the snippet. This sum is a representation of the complexity of the answer snippet in regards to the Android type in question. In other words, complexity of the snippet is defined by the number of different FQNs that are used in the snippet. A low non-zero complexity means the snippet contains very few FQNs and is mostly made up of the FQN in question. While a high complexity means the snippet contains many different FQNs in addition to the one in question. The complexity between an Android FQN and a snippet can be modelled by the following equation:

$$c(fqn, snip) = \begin{cases} \sum tf(fqn, snip) idf(fqn, snip)_i & : present \\ 0 & : otherwise \end{cases}$$

Where i represents each FQN in the snippet. For example, given the Android type `android.location.LocationManager`, we wish to find example snippets that make use of this Android type. Figure 5 and Figure 6 are examples of snippets that contain `android.location.LocationManager`. Figure 5 has a low complexity score because the snippet contains only one call to a method in `android.location.LocationManager`. Figure 6 has a higher complexity score because the snippet contains references in addition to `android.location.LocationManager`, such as `android.location.Criteria` and `java.lang.String`.

```
locationManager.requestLocationUpdates(
                LocationManager.NETWORK_PROVIDER,
                MINIMUM_TIME_BETWEEN_UPDATES,
                MINIMUM_DISTANCE_CHANGE_FOR_UPDATES,
                new MyLocationListener());
```

Figure 5: Snippet complexity score regarding `android.location.LocationManager`: 4

```
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_HIGH);
criteria.setPowerRequirement(Criteria.POWER_HIGH);

LocationManager lm = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
String provider = lm.getBestProvider(criteria, true);
```

Figure 6: Snippet complexity score regarding `android.location.LocationManager`: 66

This scoring mechanism allows the user to rank questions on how comprehensive they wish the example to be.

6. EVALUATION

To understand the effectiveness of our tools in documenting code snippets, suggesting related Stack Overflow posts, and suggesting API usage examples, we conducted a user experiment with participants from the University of Waterloo. All participants were graduate students in the Computer Science faculty. The participants' experience levels can be seen in Table 2.

Table 2: Study participant self-reported experience levels.

Participent	Years of Java	Android API
P1	3	Beginner
P2	5	Beginner
P3	5	None
P4	<1	None

6.1 Code Understanding

To evaluate the effectiveness of documenting code snippets we asked our participants to explain the functionality of certain methods found in randomly selected Stack Overflow code snippets. We limited the pool of snippets used during the selection process to snippets that are between 3 and 15 lines long, and have at least a quarter of their lines enhanced with documentation. The purpose of the questions was to simulate a SO user that is unfamiliar with a method and therefore needs to find some way of determining what

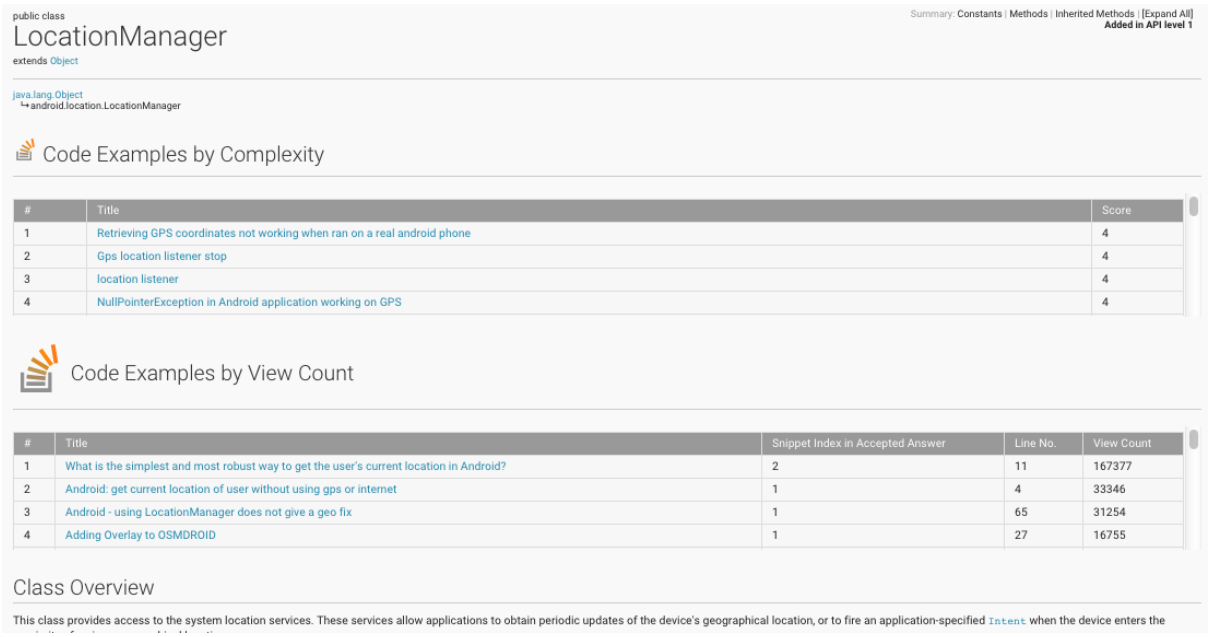

Figure 4: A screen shot of Stack Overflow examples ranked by complexity and view count, injected into official Android documentation.

it does. We discarded snippets that had fewer than a quarter of their lines enhanced as we already know the precision and recall of Baker, in other words, we are not evaluating what percentage of lines our tool can enhance, but rather how useful the enhancement is when it covers a reasonable portion of the code. The length of the snippets was limited in order to minimize the amount of time required from our participants.

Every participant was shown two accepted answers on SO, each containing a snippet. They were then asked to answer our questions about the first snippet without the use of our tool, and the questions about the second snippet with the use of our tool. In both cases they could use any other resource they desired. Snippets were assigned to participants as shown in Table 3. Although it isn't possible to let a user analyze the same snippet with, and without the tool, since they will have already answered the questions, the assignment scheme does allow each snippet to be analyzed both with, and without the tool, by different participants. For instance, participant P1 analyzed snippet S1 without the tool but participant P2 analyzed S1 with the tool.None of the participants could answer our questions (e.g., What does the `setAdapter` method do?) without consulting additional resources.

We made two observations, the first is a trend in participants' search patterns when not using the tool; they all searched the type that the method was being called on first. As an example, participant P3 was presented with the snippet in Figure 7, and asked to explain the method `setEmptyView`. To do this, he googled the class `ListView`, the first result was the official Android documentation for `ListView` which he opened up. He then used Google Chrome's search feature to search for the text `setEmptyView` which led him to the correct documentation for the method. In contrast, participant P4 clicked on the line containing the method, which immediately presented him with the correct documentation from which he then derived the answer "it looks like it sets the view to show when the list is empty". Although P4 found an answer quickly, he still felt the need to click the link to the Android documentation which is provided in the tool's popup in order to continue his investigation. Three of the participants clicked thorough to the main documentation, and two expressed a lack of trust in the tool (e.g., "how can I be sure the tool found the correct documentation?"). This first observation indicates the tool's ability speed up the search process, but that users need to be convinced of the tool's accuracy, which is an aspect we did not explain before the experiment.

Our second observation is that most participants not using the tool did not struggle to find the fully qualified method name, in other words, they did not confuse two methods with the same name but on different classes. The exception is participant P2 who gave up searching for the `setAdapter` method because he could not be sure of the type of the variable it was being called on (i.e., the type of the variable `answer` in `answer.setAdapter(answersList)` was not present in the snippet). With a larger number of participants we may have found more instances of confusion in non-tool users that can be solved easily with the tool's ability to find the exact fully qualified name or, at least, to find the list of possible matches.

```
ListView lv = (ListView)findViewById(android.R.id.list);
TextView emptyText = (TextView)findViewById(android.R.id.empty);
lv.setEmptyView(emptyText);
```

Figure 7: Code Snippet S4

6.2 Similarity Rating

In order to evaluate how well the Stack Overflow questions could be related using FQNs, we asked each of the partici-

Table 3: Assignment of snippets to participants.

	No Tool	Tool
P1	S1	S2
P2	S2	S1
P3	S4	S3
P4	S3	S4

pants to rank the relevancy of the top three questions that our tool suggested for a snippet. For a baseline, we also asked each participant to rank the relevancy of Stack Overflow's related questions for the same snippet. We randomly picked a snippet for each participant that followed the same criteria as the previous subsection. We asked participants to rate each related question using a Likert scale ranging from 1 (very irrelevant) to 5 (very relevant) [3].

The result is 24 Likert scale ratings from 4 snippets, each with 3 related questions provided by Stack Overflow and 3 related questions by our tool. A summary of the results is found in Table 4.

Table 4: Average Likert scale ratings for the given snippet for each participant. A 1 means the related question is very irrelevant, while 5 means the related question is very relevant.

	Stack Overflow	By FQN
P1	1	1
P2	3.33	3.66
P3	3	1
P4	2	1.66

The observations show that neither method works very well at providing what users would consider very relevant questions for the given snippets. One participant expressed that the relevant questions using both methods are really hit or miss and that the related question is either not related at all, or very related. Another participant expressed that linking related questions by FQN would generally show one line of the snippets being related, but offered no relation to the question. In other words, while the FQNs matched, the questions did not.

6.3 Example Usefulness

We evaluated the usefulness of having Stack Overflow questions embedded in the official Android documentation. Each participant was given one Android type to evaluate. We asked each participant to partially read the documentation of the Android type and briefly skim the methods. The goal of this is to emulate API understanding, as if a programmer was tasked to use the given API. We chose Android types that fit two categories, the first being very specific Android classes (i.e., not used often) and the second being very generic Android classes. The motivation for splitting the classes into categories is to see whether or not the scope of the Android type would have an influence on the results. Specific classes would be found in small subsets of Android classes, for example, `android.location.LocationManager` would only be used in Android applications that need to access the GPS service. While Generic classes, such as `android.content.Intent` is used in practically every Android application because of the Android framework depends on this class to manage content. The classes we used are shown in Table 5.

Table 5: Android types each participant was asked to evaluate regarding the usefulness of Stack Overflow snippets.

	Android Type	Category
P1	android.graphics.Rect	Specific
P2	android.location.LocationManager	Specific
P3	android.content.Intent	Generic
P4	android.widget.LinearLayout	Generic

As a baseline, each participant was asked to use Stack Overflow search to find a code snippet of their respective Android type, and rank how useful the snippet is to their understanding of that type. They were asked to rate using a Likert scale from 1 (very unhelpful) to 5 (very helpful) how useful the snippet was. We then asked each participant to use the augmented Android docs to choose one of the snippets related by views and rate the helpfulness of that snippet. The same was done in regards to snippets ranked by low complexity scores. The average results are shown in Table 6.

Table 6: Average helpfulness ratings grouped by class category.

Type	Stack Overflow Search	By View	By Complexity
Specific	3	3	2.5
Generic	2.5	2	2.5

The observations show that every method to find examples on Stack Overflow provide a somewhat helpful code example for the user's understanding. We also do not see a significant difference between Android types that are specific or generic. One participant made an insightful comment regarding this experiment: "I don't like the idea of using Stack Overflow to show examples" and suggested "using blogs instead would be better" in terms of understanding since they provide long and thoughtful explanations.

7. THREATS TO VALIDITY

There are three threats that may have influenced the validity of our evaluation and the conclusions we draw from it. The first is the number of participants in our user experiment and their homogeneous background. All four participants were graduate students in computer science, had little or no Android experience, and all but one had significant experience with Java. This limits our ability to generalize our results to other developer populations.

The second is that the number of participants as well as the limited amount of time we had with each lead to a small number of items being analyzed (e.g., rating the relevance of related questions). The items used for these tasks may therefore have biased our results since they do not necessarily represent average Stack Overflow questions, snippets, or Android types.

The third threat is that the questions we asked for the evaluation of our first contribution may not be representative of the questions a user encounters naturally when reading a code snippet.

8. DISCUSSION

8.1 Code Understanding

During our evaluation it was clear that documenting code snippets on Stack Overflow results in time savings for the user. The documentation for a method was found with a single click rather than a web search for the type of the variable followed by a search for the method within the type's documentation. What remains to be seen is how often the documentation we extract and inject into SO is sufficient for the user to understand the type or method in question. During our evaluation user's still felt the need to navigate to the main documentation website after reading the documentation we provided. They did not, however, revise their answers after doing so which may indicate that they looked for more information due to a lack of trust in the tool. To determine the usefulness of the provided documentation we propose an experiment where participants use the tool in their day to day work. We can then record metrics such as how often a user clicks through to the main documentation and the number of lines that are clicked on per snippet. This will provide a more accurate view of the benefits of the tool when used for real information needs rather than for answering our artificial questions.

Whether or not documenting code snippets results in a more accurate understanding remains unclear. Only one question received a better answer from a tool user than a non-tool user and it is unclear how often users confuse ambiguously named types and methods, which is the main advantage of our tool in terms of correctness of snippet understanding. Although Dagenais et al. [2] determined that 89% of methods are ambiguously named we do not know how often users confuse these in practice, especially when they already know that, for instance, they are looking at a snippet of Android code.

8.2 Similarity Rating

The results show that relating questions by similar FQNs found in code snippets provides little utility to the user. The approach to finding related questions in general is a complicated problem. A quick scan of our dataset to determine the relevancy of the top three questions based on FQN provided mixed results. The related questions are either very related or completely unrelated. For example, given a question on GPS, the top three questions the tool found were related, as can be seen in Table 7. However, given a question on the Android icon status bar, the tool did not provide very relevant suggestions, as seen in Table 8.

Table 7: A positive example of related questions offered by the tool.

Question	GPS not update location after close and reopen app on android
SO Suggestion 1	Close/hide the Android Soft Keyboard
SO Suggestion 2	How do I get the current GPS location programmatically in Android?
SO Suggestion 3	Android mock location on device?
Tool Suggestion 1	GPS Locater not working
Tool Suggestion 2	How can I obtain the current GPS location?
Tool Suggestion 3	Best error handling approach for GPS lookup in Android?

Table 8: A negative example of related questions offered by the tool.

Question	Icon in the status bar when application is running
SO Suggestion 1	Android status bar icon design
SO Suggestion 2	Set status bar icon names
SO Suggestion 3	Set icon for android application
Tool Suggestion 1	Can't get focus on editText when children are added in layout
Tool Suggestion 2	Forcing the default Android soft-keyboard to show
Tool Suggestion 3	How does one animate the Android sync status icon?

It appears that relating questions purely on the FQNs in the answer snippet does not provide consistently good question relations. However, it does offer some utility that is sometimes at par with Stack Overflow's suggested questions. It is also observed that when the tool does suggest good results, they are often different than Stack Overflow's suggestions. This provides an opportunity to enhance Stack Overflow's existing algorithm to make use of FQNs in their search terms.

The poor results in suggested questions can be partially attributed to Stack Overflow's user guidelines. It is generally encouraged to search the question on Stack Overflow before submitting one in order to avoid duplicates. This eliminates a lot of similarity between questions which would have an effect on both Stack Overflows related questions as well as ours.

8.3 Example Usefulness

The results show that using Stack Overflow snippets as examples of Android API types provide mediocre results in

all three of our test cases. Searching by API type on Stack Overflow seems like too broad of a search term to use to find learning examples. In addition, our tool rankings by view count often lead to snippets where the API being examined played an insignificant role in the snippet. We expected better results when matching by low complexity as this ensured that the majority of the snippet was made up of the API type. But sometimes these examples were not helpful for API understanding, but rather for troubleshooting very specific issues.

One participant did not like our approach and commented that they prefer using blogs or tutorial websites instead of Stack Overflow snippets when learning new APIs. This provided good criticism of our approach and in retrospect makes a lot of sense. Although Stack Overflow does provide snippets that can be used to learn, it is mostly a troubleshooting website. Tutorials and blogs are more oriented towards learning while Stack Overflow is more oriented towards problem solving. It would be worth adjusting Baker to crawl tutorials and blogs for snippets and using those snippets to find useful examples.

9. CONCLUSION

We used FQNs extracted from code snippets in order to improve code understanding and API discoverability in Stack Overflow and Android documentation. We found that our tool speeds up the search process that programmers under take when trying to comprehend code snippets. In addition, our tool shows the potential to solve type ambiguity issues that users face when examining code snippets. We investigated an approach to relating questions by extracted FQNs in code snippets, and we found that although the approach does not outperform existing algorithms, it may offer an added dimension to enhance existing full-text search algorithms. We attempted to improve existing documentation by linking to external code snippets. This does provide an added convenience to the user, but we did not find it a significant improvement over a simple search for the API. Nonetheless, we introduced a framework that can measure the level of detail a code snippet has in relation to a given API.

10. ACKNOWLEDGMENTS

The authors would like to thank Siddharth Subramanian for his help in obtaining the dataset and understanding the Baker API. They would also like to thank the graduate students at the University of Waterloo who participated in their user study.

11. REFERENCES

[1] Locationmanager | android developers. `http://developer.android.com/reference/android/location/LocationManager.html`. Accessed: 2015-03-20.

[2] B. Dagenais and M. P. Robillard. Recovering traceability links between an api and its learning resources. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 47–57. IEEE, 2012.

[3] R. Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[4] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[5] S. G. McLellan, A. W. Roesler, J. T. Tempest, and C. I. Spinuzzi. Building more usable apis. *IEEE software*, 15(3):78–86, 1998.

[6] J. Singer. Practices of software maintenance. In *Software Maintenance, 1998. Proceedings., International Conference on*, pages 139–145. IEEE, 1998.

[7] Stack Overflow related questions algorithm. `http://stackoverflow.com/questions/891772/stackoverflow-related-questions-algorithm`. Accessed: 2015-03-20.

[8] S. Subramanian and R. Holmes. Making sense of online code snippets. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 85–88. IEEE Press, 2013.

[9] S. Subramanian, L. Inozemtseva, and R. Holmes. Live api documentation. In *Proceedings of the 36th International Conference on Software Engineering*, pages 643–652. ACM, 2014.