NLP2Code: Code Snippet Content Assist via Natural Language Tasks

Brock Angus Campbell and Christoph Treude
School of Computer Science
University of Adelaide
Adelaide, SA, Australia
a1687816@student.adelaide.edu.au, christoph.treude@adelaide.edu.au

***Abstract*—Developers increasingly take to the Internet for code snippets to integrate into their programs. To save developers the time required to switch from their development environments to a web browser in the quest for a suitable code snippet, we introduce NLP2Code, a content assist for code snippets. Unlike related tools, NLP2Code integrates directly into the source code editor and provides developers with a content assist feature to close the vocabulary gap between developers' needs and code snippet meta data. Our preliminary evaluation of NLP2Code shows that the majority of invocations lead to code snippets rated as helpful by users and that the tool is able to support a wide range of tasks.**

Video: https://www.youtube.com/watch?v=h-gaVYtCznI

I. Introduction and Motivation

The Internet has provided software developers with an effective infrastructure for sharing and accessing programming knowledge, often curated through social media mechanisms, such as voting or commenting [1]. The prime example of this is the Question and Answer website Stack Overflow[1] with more than 14 million questions and more than 22 millions answers as of July 2017. Many of the questions and answers contain code snippets [2], and much research has been conducted on the quality of these snippets (e.g., [3], [4]).

Question and Answer websites are not primarily designed for direct code reuse [5]. With the current tooling available, a common scenario is for a developer to write code in an Integrated Development Environment (IDE), reach a point at which help is needed, switch to a web browser, type a query into a search engine, evaluate search results, find a suitable solution, and integrate the newly acquired knowledge into the source code in the IDE—often in the form of a code snippet.

Following the argument that too many context switches between different tools can significantly impact the productivity of a developer [6], we introduce NLP2Code, a plugin for the Eclipse IDE, which allows developers to integrate code snippets from Stack Overflow into their code without leaving the source code editor. Unlike previous efforts to integrate Stack Overflow content into the IDE, such as Seahawk [7], Prompter [8], and T2API [9], NLP2Code does not introduce additional views or windows to the IDE, but works solely in the source code editor.

[1]http://stackoverflow.com/

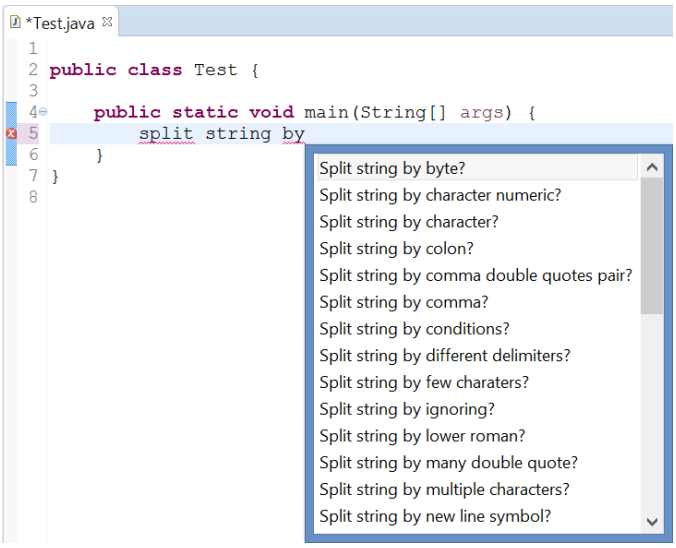

Fig. 1. NLP2Code's content assist

Similar to Eclipse's code completion mechanism [10], we introduce a content assist for code snippets. This content assist is aimed at closing the vocabulary gap [11] between developers' needs and relevant Stack Overflow threads, making suggestions based on natural language tasks that we extract from Stack Overflow thread titles, using our previous work on extracting tasks from software documentation [12], [13].

Our preliminary evaluation of NLP2Code with 6 participants and 101 invocations showed that the majority of invocations led to helpful snippets, often based on the first result, and that NLP2Code's responses to these queries outperformed the state of the art. Participants used NLP2Code for a wide range of tasks from looking up implementations of particular algorithms to finding API usage examples and template code.

Compared to previous work, we make three contributions:

- a code snippet content assist feature, designed to close the vocabulary gap between developers' needs and code snippet meta data,
- seamless integration of code snippets from Stack Overflow into code without having to leave the source code editor, and
- a preliminary evaluation with 101 queries, including a comparison to the state of the art.

Fig. 2. NLP2Code's flow. To transition from (a) to (b), select a content assist suggestion, and to cycle through code snippets, such as (c), press `Ctrl+'`

II. NLP2Code

We developed NLP2Code as a plugin to the Eclipse IDE. However, none of the underlying concepts are Eclipse or Java-specific, and we only focused on this ecosystem since we had access to undergraduate students working with Eclipse for a preliminary evaluation and since Eclipse offers a plugin infrastructure that has already benefited many other research projects (e.g., [10]).

After installing the NLP2Code plugin, a user can open its content assist anywhere in the source code editor by pressing `Ctrl+1`.[2] Figure 1 shows a screenshot of the content assist after typing *"split string by"*.[3] As shown in this example, even if a user does not exactly know how to specify what they want to split strings by, the content assist helps close this vocabulary gap by providing a large number of suggestions for which NLP2Code can provide code snippets. The content assist suggestions are the 599,550 natural language tasks that the TaskNav algorithm [12], [13] extracted from the 1,109,677 Stack Overflow thread titles tagged "java" in the latest Stack Overflow data dump.[4]

The TaskNav algorithm conceptualizes tasks in software documentation as verbs associated with a direct object and/or a prepositional phrase, such as *"get iterator"*, *"get iterator for collection"*, and *"add to collection"*. After removing markup and pre-processing code elements and incomplete sentences (see [12] for details), the algorithm makes use of the grammatical dependencies—relations between words in a sentence as identified by the Stanford NLP toolkit [14]. Because tasks can be described in different grammatical ways (e.g., *"returning an iterator"*, *"return iterator"*, *"iterator returned"*, and *"iterator is returned"*), dependencies between words in active and passive voice are considered. In addition, context might be important, e.g., whether an iterator is returned or whether the documentation instructs the user to *"not return iterator"*. Furthermore, the iterator might be specified using additional words, such as *"list iterator"*, and prepositional phrases might make the task description more specific, such as *"return iterator of collection"*. For example, TaskNav extracted the task *"add lines to text file"* from the Stack Overflow question title *"Best strategy to add lines of text to a text file"*.[5] TaskNav uses a handcrafted list of about 200 programming actions and about 30 generic objects to filter out tasks irrelevant to software development.

Our assumption that developers need assistance specifically for tasks is supported by the benchmark used to evaluate SWIM [15] where 25 of the 30 benchmark queries follow a similar task pattern. In addition to providing content assist suggestions, the task-based approach helps overcome the limitations of the commonly used bag-of-words assumption and can distinguish between tasks such as *"convert int to string"* and *"convert string to int"* [16].

Content assist suggestions in NLP2Code can be filtered by typing characters and words, and pressing `Enter` selects a suitable suggestion. After the selection, NLP2Code conducts a query using the Google Custom Search Engine[6] on http://stackoverflow.com/. The plugin then collects up to three code snippets (i.e., content surrounded by `<pre><code>` tags) from answers with the highest scores from the four first

[2] All key bindings are configurable.

[3] *"split string by whitespaces"* was one of the tasks that our study participants completed using the NLP2Code content assist.

[4] https://archive.org/details/stackexchange, up to 12 tasks per title.

[5] http://stackoverflow.com/q/26575009

[6] https://cse.google.com.au/cse/

TABLE I
HELPFUL AND UNHELPFUL NLP2CODE INVOCATIONS

	NLP2Code	T2API
correct/helpful	74	16
incorrect/unhelpful	27	45
no code snippet	0	40

Stack Overflow threads returned by the search. We chose these numbers by trading off performance (impacted by website queries) and diversity of results (by considering different threads). The code snippet from the answer with the highest score in the first thread is automatically inserted into the source code editor along with a comment indicating its source, replacing the original text. The user can then cycle through different code snippets using `Ctrl+`.

Figure 2 shows an example of this workflow from our preliminary evaluation. In this case, the participant required help with *"add lines to text file"* and NLP2Code assisted with the suggestions shown in Figure 2(a) after the user typed *"add lines"*. Figure 2(b) shows the first code snippet that NLP2Code returned along with the comment indicating its source. In this case, the participant already rated the first code snippet as helpful. Another option would have been to cycle through further code snippets using `Ctrl+`. Figure 2(c) shows the second snippet that NLP2Code would have returned.

In addition to the content assist described above, NLP2Code can be invoked by highlighting any text in the Eclipse IDE and pressing `Ctrl+6` or by surrounding text with question marks (e.g., *"?add lines to text file?"*).

III. EXAMPLE SCENARIOS

In this section, we discuss two example scenarios for developers using NLP2Code.

a) API usage: One scenario for the use of NLP2Code is that of learning how to use an API. As an example, one of our study participants needed help with *"add custom jpanel to jframe"*. For this task, the participant selected the code snippet in Stack Overflow answer 22621494[7], which was the second code snippet that NLP2Code returned. The code snippet consists of ten lines of code, first setting up instances of `JFrame` and `JPanel` before adding the latter to the former. This code snippet was rated as helpful by the participant.

b) Algorithms: Another group of tasks that our participants completed with NLP2Code is the reuse of algorithms. One of our study participants needed help with *"complete bubble sort"*. The first code snippet returned by NLP2Code came from Stack Overflow answer 16089042[8], and it consists of a method implementing bubble sort in 13 lines of code. The participant rated this snippet as helpful.

IV. PRELIMINARY EVALUATION

As a preliminary evaluation of the potential of NLP2Code, we asked ten undergraduate students at the University of Adelaide who are not connected to this research project to use the plugin as part of the development for their coursework for two weeks. In addition to the tasks that they used to invoke NLP2Code, the plugin recorded whether their task came from content assist, how many code snippets they cycled through, and whether they considered the final code snippet to be helpful. For the latter, we asked them at the end of each invocation (i.e., at the first `Enter` keypress after an invocation) *"Was this code snippet helpful?"*. In addition, we inserted all of their queries into the state-of-the-art tool T2API, a statistical machine translation tool that takes a given English description of a programming task as a query and synthesizes an API usage template for this task [9], and we compared the results to those produced by NLP2Code.

[7]http://stackoverflow.com/a/22621494
[8]http://stackoverflow.com/a/16089042

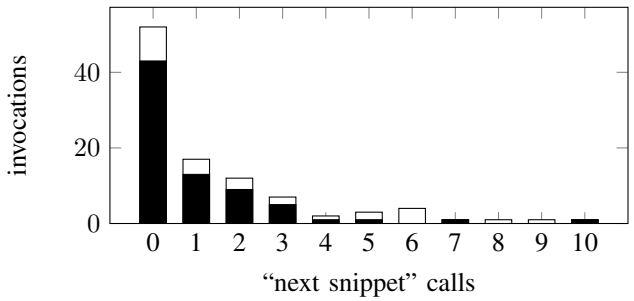

Fig. 3. Number of "next snippet" calls; black denotes helpful snippets and white denotes unhelpful snippets

TABLE II
INVOCATIONS WITH CONTENT ASSIST (EXCERPT)

implement dijkstra algorithm
complete bubble sort
sort array for binary search
convert uppercase to lowercase
add custom jpanel to jframe
call sleep on current thread
find number of shortest paths
implement depth first search
compile basic hello world program
generate random integers

A. Evaluation with student participants

At the time of our study, all participants were undertaking the course "Algorithm & Data Structure Analysis" at the University of Adelaide in which they were required to use Java

TABLE III
INVOCATIONS WITHOUT CONTENT ASSIST (EXCERPT)

initialize two dimensional vector
round double value to two decimal places
download html from a website
remove html tags from a string
evaluate math expression in string form
play a sound
calculate difference between two date instances
generate all permutations of an arraylist
implement floyd warshall
change seed of random generator

for their course assignments. We provided all participants with instructions on how to use NLP2Code and encouraged them to use the plugin as part of their Java development. Six of the undergraduate students used NLP2Code and shared their usage data with us (response rate 60%).

The first two columns of Table I show the number of invocations that resulted in helpful and unhelpful code snippets, respectively. Our six participants used NLP2Code for 101 invocations, the majority of which resulted in a helpful code snippet. We did not filter out any invocations for the analysis. For about half of the invocations, the content came from content assist. Table II shows examples of invocations using content assist, and Table III shows invocations without using content assist, i.e., invocations made by highlighting text and pressing `Ctrl+6` or by surrounding text with question marks. Note that even the invocations without the content assist feature usually follow a task pattern, i.e., starting with a verb followed by a noun phrase, such as *"initialize two dimensional vector"*. These invocations suggest that our underlying assumption that developers need support for tasks is warranted. Our future work will investigate how we can expand NLP2Code's content assist feature to support more of these tasks.

Figure 3 shows the number of "next snippet" calls that our participants made using `Ctrl+`` and how many of these invocations led to a helpful result. In many of the cases, the participants did not cycle through results, but chose the first code snippet returned by NLP2Code. The majority of these snippets was seen as helpful. Other invocations resulted in up to ten "next snippet" calls, with similar success rates. We conclude that often, the first code snippet returned by NLP2Code can solve the user's task, and in cases where this does not happen, the "next snippet" feature has a high chance of helping the user.

B. Comparison with state of the art

NLP2Code is most similar to the recently published T2API [9]. In contrast to NLP2Code, T2API has no content assist, its integration into the IDE is not seamless since queries are made in a separate window, and the authors have not evaluated the tool with users. As the name suggests, T2API is specific to API calls whereas NLP2Code can provide code snippets for any task documented in the answers on Stack Overflow. For example, our preliminary evaluation suggests that providing help with algorithms is one of the major benefits of NLP2Code.

When we inserted the 101 queries that our participants produced into T2API,[9] only 16 (i.e., 16%) resulted in reasonable code snippets, see the last column of Table I. 40 queries did not return any code snippet, and the remaining 45 resulted in obviously incorrect code snippets. For example, the query *"convert inputstream to string?"* resulted in the following code snippet from T2API: `String.split(); String.length(); StringBuilder.toString(); String.substring();`, whereas NLP2Code produced `String myString = IOUtils.toString (myInputStream, "UTF-8");`.[10] While these findings might not generalize to other settings (e.g., outside of the university environment), we conclude that at least for the kind of queries produced by our participants, NLP2Code outperforms T2API. This is likely explained by the fact that T2API's focus is on API suggestions, not code suggestions.

V. Related Work

Similar tools to T2API that also do not provide content assist include the Bing Developer Assistant [17], DeepAPI [16], and SWIM [15].

Other prominent tools that integrate Stack Overflow content into the IDE are Seahawk [7] and Prompter [8]. Seahawk formulates queries automatically based on the active context of an IDE, presents a ranked list of results, and lets users import code snippets through drag and drop. Prompter takes this idea a step further by notifying developers about the available help. Similar to T2API, these two tools utilize additional windows or views in the IDE and do not integrate into a source code editor in the same way that NLP2Code does. In addition, Seahawk and Prompter do not allow users to search for code snippets for a given task.

In the area of code snippet search, Zagalsky et al. [18] presented Example Overflow, an online code search and recommendation tool. Unlike NLP2Code, Example Overflow is not integrated into an IDE, but is its own website. In their work on making sense of online code snippets, Subramanian and Holmes [19] extracted structural information from short plain-text snippets on Stack Overflow and showed that these structural relationships could improve code snippet search.

VI. Conclusion and Future Work

We introduced NLP2Code, a content assist for code snippets integrated into the Eclipse IDE. The goal of NLP2Code is to eliminate the context switch that occurs when developers look up code snippets in a web browser while developing in an IDE. NLP2Code closes the vocabulary gap between developers' needs and code snippet meta data through a content assist feature based on natural language tasks extracted from the titles of questions on Stack Overflow. In our preliminary evaluation, NLP2Code returned helpful code snippets in the majority of cases while supporting a wide range of tasks, including API usage examples, template code, and algorithms.

Based on these encouraging results, our future work includes further improvements to NLP2Code's content assist feature and to the mapping of tasks to code snippets (e.g., avoiding outdated code snippets, selecting those that integrate easily into the existing code base, selecting those that are easy to understand [20], or adding additional documentation on-demand [21]), as well as the automated integration of code snippets into the existing code base (e.g., by building on Jigsaw [22]). In addition, we plan to evaluate the tool with professional developers in an industry setting.

[9] http://atwood.encs.concordia.ca:5905/T2APIRESTService/

[10] http://stackoverflow.com/a/350723

ACKNOWLEDGEMENTS

We thank the students who participated in our evaluation for using NLP2Code as part of their studies.

REFERENCES

[1] C. Treude, F. Figueira Filho, B. Cleary, and M.-A. Storey, "Programming in a socially networked world: the evolution of the social programmer," in *Proceedings of the CSCW Workshop on the Future of Collaborative Software Development*, 2012, pp. 1–3.

[2] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web? (NIER track)," in *Proceedings of the International Conference on Software Engineering*, 2011, pp. 804–807.

[3] F. Calefato, F. Lanubile, M. C. Marasciulo, and N. Novielli, "Mining successful answers in Stack Overflow," in *Proceedings of the Working Conference on Mining Software Repositories*, 2015, pp. 430–433.

[4] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example?: A study of programming Q&A in StackOverflow," in *Proceedings of the International Conference on Software Maintenance*, 2012, pp. 25–34.

[5] O. Barzilay, C. Treude, and A. Zagalsky, "Facilitating crowd sourced software engineering via Stack Overflow," in *Finding Source Code on the Web for Remix and Reuse*, S. E. Sim and R. Gallardo-Valencia, Eds. Springer, 2013.

[6] S. Proksch, V. Bauer, and G. C. Murphy, "How to build a recommendation system for software engineering," in *Software Engineering: International Summer Schools, LASER 2013–2014, Elba, Italy, Revised Tutorial Lectures*, B. Meyer and M. Nordio, Eds. Springer International Publishing, 2015.

[7] L. Ponzanelli, A. Bacchelli, and M. Lanza, "Seahawk: Stack Overflow in the IDE," in *Proceedings of the International Conference on Software Engineering*, 2013, pp. 1295–1298.

[8] L. Ponzanelli, G. Bavota, M. D. Penta, R. Oliveto, and M. Lanza, "Prompter: A self-confident recommender system," in *Proceedings of the International Conference on Software Maintenance and Evolution*, 2014, pp. 577–580.

[9] T. Nguyen, P. C. Rigby, A. T. Nguyen, M. Karanfil, and T. N. Nguyen, "T2API: Synthesizing API code usage templates from English texts with statistical translation," in *Proceedings of the International Symposium on the Foundations of Software Engineering*, 2016, pp. 1013–1017.

[10] M. Bruch, M. Monperrus, and M. Mezini, "Learning from examples to improve code completion systems," in *Proceedings of the Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering*, 2009, pp. 213–222.

[11] P. Mika, E. Meij, and H. Zaragoza, "Investigating the semantic gap through query log analysis," in *Proceedings of the International Semantic Web Conference*, 2009, pp. 441–455.

[12] C. Treude, M. P. Robillard, and B. Dagenais, "Extracting development tasks to navigate software documentation," *IEEE Transactions on Software Engineering*, vol. 41, no. 6, pp. 565–581, 2015.

[13] C. Treude, M. Sicard, M. Klocke, and M. P. Robillard, "TaskNav: Task-based navigation of software documentation," in *Proceedings of the International Conference on Software Engineering - Volume 2*, 2015, pp. 649–652.

[14] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics System Demonstrations*, 2014, pp. 55–60.

[15] M. Raghothaman, Y. Wei, and Y. Hamadi, "Swim: Synthesizing what I mean: Code search and idiomatic snippet synthesis," in *Proceedings of the International Conference on Software Engineering*, 2016, pp. 357–367.

[16] X. Gu, H. Zhang, D. Zhang, and S. Kim, "Deep API learning," in *Proceedings of the International Symposium on the Foundations of Software Engineering*, 2016, pp. 631–642.

[17] H. Zhang, A. Jain, G. Khandelwal, C. Kaushik, S. Ge, and W. Hu, "Bing developer assistant: Improving developer productivity by recommending sample code," in *Proceedings of the International Symposium on the Foundations of Software Engineering*, 2016, pp. 956–961.

[18] A. Zagalsky, O. Barzilay, and A. Yehudai, "Example Overflow: Using social media for code recommendation," in *Proceedings of the International Workshop on Recommendation Systems for Software Engineering*, 2012, pp. 38–42.

[19] S. Subramanian and R. Holmes, "Making sense of online code snippets," in *Proceedings of the Working Conference on Mining Software Repositories*, 2013, pp. 85–88.

[20] C. Treude and M. P. Robillard, "Understanding Stack Overflow code fragments," in *Proceedings of the International Conference on Software Maintenance and Evolution*, 2017, to appear.

[21] M. P. Robillard, A. Marcus, C. Treude, G. Bavota, O. Chaparro, N. Ernst, M. A. Gerosa, M. Godfrey, M. Lanza, M. Linares-Vásquez, G. Murphy, L. Moreno, D. Shepherd, and E. Wong, "On-demand developer documentation," in *Proceedings of the International Conference on Software Maintenance and Evolution*, 2017, to appear.

[22] R. Cottrell, R. J. Walker, and J. Denzinger, "Jigsaw: A tool for the small-scale reuse of source code," in *Companion of the International Conference on Software Engineering*, 2008, pp. 933–934.