Example Overflow: Using Social Media for Code Recommendation

Alexey Zagalsky
Blavatnik School of Computer Science
Tel-Aviv University
Tel-Aviv, Israel
alexeyza@tau.ac.il

Ohad Barzilay
Blavatnik School of Computer Science
Tel-Aviv University
Tel-Aviv, Israel
ohadbr@tau.ac.il

Amiram Yehudai
Blavatnik School of Computer Science
Tel-Aviv University
Tel-Aviv, Israel
amiramy@tau.ac.il

***Abstract*—Modern Q&A websites, such as Stack Overflow, use social media to provide concise answers, and offer rich technical context with quality assessment capabilities. Although some of the answers may include executable code snippets, they are entangled in free text and are not easily extracted. Q&A websites are not designed for such direct code reuse.**

We present Example Overflow, a code search and recommendation tool which brings together social media and code recommendation systems. Example Overflow enables crowd-sourced software development by utilizing both textual and social information, which accompany source code on the Web. Its browsing mechanism minimizes the context switch associated with other code search tools. In this paper we describe the development of the tool, provide preliminary evaluation, and discuss its contribution to an example centric programming paradigm.

Keywords-code search; example overflow; stack overflow; code repository; example embedding; example centric programming; crowd-sourced software development; social recommendations;

I. INTRODUCTION

Social media provides useful recommendations for many areas of our lives. For example, when considering what movies to watch, one may use recommendations from his or her immediate social cycle (e.g. Facebook friends), or use the wisdom of the crowd [20], using, for instance, the ratings on imdb.com. This is part of a more general trend in which social recommendations (e.g. Facebook) have begun to replace search (e.g. Google Search).

The Software Engineering (SE) domain is no different; social media has been shown to be beneficial in many areas of SE including feature prioritization [1], risk analysis [19], collaborative filtering [7], knowledge management [9], and documentation [4] [22].

In this paper we introduce a new application for social recommendations. We present Example Overflow - a code search and recommendation tool - which leverages the body of knowledge created by the socio-professional media, to recommend high quality, embeddable code. Our tool uses built-in social mechanisms of the popular Q&A website, Stack Overflow[1]. To the best of our knowledge, at present, there is no general-purpose software development crowd-sourcing platform [24].

Example Overflow is a live system, and is currently deployed as a public and free website [2]. Our initial implementation contains all code snippets that appear in accepted jQuery related answers (more than 33,000 code snippets). jQuery[3] is a popular JavaScript library, initially released in 2006 and is ranked fifth in its popularity on Stack Overflow (with over 150,000 related questions). We chose it as our case study since we assume that Web developers would find it easier to adopt an example centric programming approach. Our decision is also supported by the following: (1) Parnin and Truede [17] found that Stack Overflow covers 84.4% of the jQuery API, and (2) Our study shows that 20% of the jQuery related questions have a code snippet embedded in their accepted answer.

Example Overflow is developed as part of a comprehensive effort to create an Example Embedding Ecosystem [2] – an example centric development method in which example related concerns are weaved in the development process, software tools, practices, training, organization culture and more.

Software development crowd sourcing, as manifested by our approach, is challenging the division of labor between the human and the machine [23]. In many existing code search tools, such as Krugle[4] and Koders[5], the machine is in charge of evaluating the quality and relevance of the code found. In Example Overflow, on the other hand, humans, i.e. the socio-professional community, are assessing the code, and the machine only facilitates the process of example embedding.

The remainder of this paper is structured as follows. In section II we review related work by considering the alternatives offered by other code search tools. In Section III we describe Stack Overflow on which we base our tool, and in Section IV we briefly elaborate on our design decisions. In Sections V, VI and VII we examine some of the aspects involved in the development of Example Overflow, provide preliminary evaluation, and discuss its advantages with respect to Stack Overflow. In section VIII, we elaborate on the limitations of example centric programming in general

[1] http://www.youtube.com/watch?v=NWHfY lvKIQ

[2] http://www.exampleoverflow.net/

[3] http://jquery.com/

[4] http://www.krugle.com/

[5] http://www.koders.com/

and of crowd sourced software development in particular, and we discuss threats to validity. Finally, in section IX we outline our future work.

II. Related Work

Programming by example was found to be intuitive to many developers, novices and experts alike [13]. Tools such as Strathcona [11] and PARSEWeb [21] provide developers with code fragment recommendations, taken from a central code repository, by generating queries based on code context and the structural details of the developer's activity. The quality of the code found by these tools is derived from the overall quality of the repositories they use.

Code search engines, on the other hand, such as Krugle[6] and Koders[7], search in a large set of open source repositories, but do not provide explicit mechanisms to evaluate or improve the quality of the found snippets. Other tools like MICA [18], Exemplar [6] or [15] use API calls or API examples to recommend example code, but they are restricted to providing a limited set of examples based on the API only.

Using social media, however, allows Example Overflow to scale beyond specific code repositories and to leverage human brainpower [23] to assess the quality of specific code snippets.

Some tools like [12] use a tool specific query language in order to improve their search precision. Example Overflow, however, takes the 'Google Paradigm', allowing natural language search queries. This approach is also advocated by [18].

III. Stack Overflow

Stack Overflow[8] uses social mechanisms to facilitate knowledge exchange between users and to create an information archive. In Stack Overflow, a programmer can ask a question about almost any programming related topic, and receive a detailed response within 10 minutes median [14]. Answers on Stack Overflow often become a substitute for official product documentation, when the official documentation is sparse or currently non-existent[9]. Truede *et al.* [22] found *code review* to be one of the most effective usages of Stack Overflow.

The way Stack Overflow is designed allows each question and answer to be rated. Eventually for each question, the best answer is chosen to be "the accepted answer" for that question. In addition, members can edit each question and each answer to allow the information to constantly evolve and remain up to date. Finally, Stack Overflow has an enormous community of members, it is an already big knowledge base (currently it has over 2.6 million questions and 1.6 million accepted answers) and it is constantly growing.

[6] http://www.krugle.com/
[7] http://www.koders.com/
[8] www.stackoverflow.com
[9] https://stackoverflow.fogbugz.com/default.asp?W25450

IV. Design Decisions

In order to implement a crowd sourced software recommendation system, one needs to explicitly foresee the division of labor between the developer and the machine. The system should facilitate the core practices involved in Example Embedding namely enable browsing and comparing multiple code examples and minimizing the developer's context switch as elaborated below.

A. Comparing Multiple Examples

Experienced developers, with whom we discussed example centric development, reported browsing multiple examples, comparing them side by side, and eventually choosing the examples most suitable for the developer's needs (sometimes merging multiple examples). This also conforms to the literature suggesting that it is easy to extract the repetitive example structure from a specific context, and to reuse the repetitive part for new tasks [16][10].

Traditional code search tools (e.g. Google Code Search, Krugle) allow searching for code, where a developer inputs a query and then he or she is displayed with the search results consisting of the filename or the first few lines of the source code. The developer is then forced to click on each result, open it in a new view, inspect it separately and decide whether it is the best example to be found. Using these tools, there is no way for the developer to compare the current code example with the ones viewed previously or the one to be inspected next.

When searching in Example Overflow, on the other hand, the developer is presented with the code of the 5 most suitable results. Our preliminary evaluation supports this decision as can be seen in section VI. This allows the developer to see all the code examples in the same view, where they are not isolated from each other, compare them and choose the one that suites him or her best. If none of the results are suitable, then automatically the next 5 most suitable code examples are displayed as well. This way the developer will be presented with the minimum amount of code examples that are needed to find the most suited one(s).

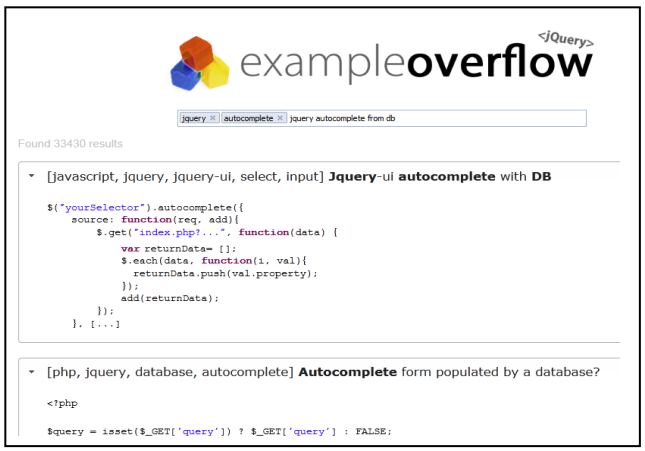

Figure 1. Example Overflow Web interface.

B. Minimal Context Swithcing

We argue that example search is an integral part of modern software development (Example Embedding Ecosystem [2]). Therefore we aim to allow the developer to find example code with minimal context switching as possible, ideally without leaving the IDE.

To support this, our design has a single search window, as can be seen in Figure 1. The developer is presented with the best search results, where each result shows only the code snippet itself, thus allowing the developer to see all the code snippets at the same view without opening new views. If the developer needs more context for the code snippet, all he or she has to do is hover (without even clicking) over the example with the mouse, and choose either "Question" or "Answer" to see that context inside the same view.

V. EXAMPLE OVERFLOW

A. Populating the Repository

We use Stack Overflow's API to request all the questions relevant to our current domain, jQuery tagged questions, where we filter out all the questions without an accepted answer. We follow a conservative approach by choosing only accepted answers to ensure retrieval of high quality results. The next step is to check whether each of these questions has a code snippet inside the accepted answer. If so, that code snippet is extracted and saved to our database with all the accompanying information: the question title, the question body, the answer body, the code snippet itself, the user rating of the answer from Stack Overflow, the view count of the question, the tags associated with the question and other relevant information. If that question is already in our database we only update the changed information. This process can be executed as a scheduled task to allow us to keep the data in sync with the data at Stack Overflow.

B. Searching

Example Overflow uses keyword search based on the Apache Lucene [8] library, which internally uses the term frequency-inverse document frequency (tf-idf) weight [25]. In order for Apache Lucene to search, one needs to define which parameters are to be analyzed and indexed. For keyword search index we use both the code snippet and the additional metadata which accompanied the code snippet at Stack Overflow. This allows a developer to find code snippets that may not contain the search query keyword, but the keyword appears in the contextual data and indicates that it has been used in that context.

Each code example is represented as a document with several parts: title, tag, answer, question, code, and social metadata. We use the following formula to calculate the score of each document representing a code example:

$$S_{doc} = [\ W_{title}S_{title} + W_{tag}S_{tag} + W_{answer}S_{answer} + W_{question}S_{question} + W_{code}S_{code}\ ]\ S_{metadata} \quad (1)$$

Where each S_{part} represents the individual score of the respective part of the document, and W_{part} represents the weights that may be chosen to tune the tool for the best results possible. The weights would be computed based on a set of experiments, but for the initial evaluation presented in this paper, they were chosen heuristically to give higher priority to results with matching keywords in the title or tag, over matches in the other parts, and are $W_{title}=4$, $W_{tag}=4$, $W_{answer}=1$, $W_{question}=1$, $W_{code}=2$.

VI. PRELIMINARY EVALUATION

As a preliminary evaluation, we used a jQuery benchmark to compare the characteristics of Example Overflow with other existing code recommendation systems, as elaborated below.

A. Evaluation Setup

We used the code assignments from the book *jQuery in Action* [3] to define a benchmark of ten frequent programming tasks shown in Table I. For each task we have manually decided on a concise query to be used by a potential developer in order to find the desired code snippet. We have used the same query in each of the following tools, and have examined the first 20 results returned for each query.

TABLE I. SEARCH QUERIES USED FOR THE EVALUATION BENCHMARK

Data Point	Search Query
Dynamic Dimension	"jquery dynamic dimension"
Hover	"jquery hover div"
Position	"jquery position"
Rounded Corners	"jquery rounded corner"
Draggable	"jquery draggable"
Droppable	"jquery droppable"
Autocomplete	"jquery autocomplete from db"
Accordion	"jquery accordion"
Date Picker	"jquery datepicker"
Image Scale	"jquery image scale effect"

We used the following existing tools in the evaluation: Google Search, Stack Overflow, Krugle, and Koders. We also used Google Code Search in our preliminary evaluation, where it had similar results to Krugle, but recently this service has been shut down by Google.

We have not included Strathcona [11], Blueprint [5] or PARSEWeb [21] in the evaluation, because they are domain specific and would not work for the jQuery domain.

B. Evaluation Methodology

For each query and each tool we have received a list of results. These results were manually examined by one of the authors (see section VIII). We have used the actual code from the book *jQuery in Action* as a point of reference.

We examined the list of results retrieved from each tool, and determined whether it accomplished the programming task. If no matching result was found at the top 20 results, it was marked as "not found" and received a rank of 21 for the average calculation.

C. Evaluation Results

Table II shows the rank location of suitable example code in the search results returned from each tool.

TABLE II. SEARCH RESULTS COMPARISON: RANK OF A SUITABLE EXAMPLE AT THE RETURNED SEARCH RESULTS.

Data Point	Code Repository Tools				
	Google Search	***Krugle***	***Koders***	***Stack Overflow***	***Example Overflow***
Dynamic Dimension	4	Not found	Not found	1	3
Hover	1	2	1	1	2
Position	3	Not found	Not found	4	1
Rounded Corners	2	Not found	3	3	1
Draggable	1	Not found	3	2	1
Dropable	1	Not found	3	1	2
Autocomplete	1	Not found	Not Found	1	1
Accordion	1	Not found	12	3	1
Date Picker	1	Not found	3	1	1
Image Scale	2	Not found	Not found	Not found	3
Avg. Rank	1.7	19.1	9.7778	3.8	1.6

It can be seen that our tool has overall the best results with an average result rank of 1.6, where Google Search and Stack Overflow show similar results with 1.7 and 3.8 respectively, but Krugle and Koders have poor results. The reason for the poor results may be that both of them search for keywords to match the search query, without taking into account the context of the found keyword or any additional metadata. On the other hand our tool gives different weights to keywords based on their origin (code, title, tag, question, and answer). With this approach we obtain better results. Another possible cause for the poor results might be because both Krugle and Koders are limited to open source projects, where recent domains such as jQuery are not currently present. In addition it can be seen that our tool didn't require loading additional results (by scrolling down) and managed to show a suitable code example in the top 5 results.

During our evaluation we have also examined the amount of view/context switches by counting the number of mouse clicks required by the developer between the search request and until the developer was able to see the actual suitable example code.

It can be seen at Table III that our approach has an average of 0 mouse clicks hence it doesn't require the developer to switch views or open new views, but instead we immediately show the developer the actual relevant code snippets.

During our preliminary evaluation process we noticed that both Krugle and Koders returned results which linked to actual project files, without guiding the developer to the location of the required example code inside the project. This requires the developer to read that file as a whole, and search for the possible match for his query, thus forcing the developer to context switch from his actual task. In addition, most of their returned search results are only partial and have context in other files of that project, which requires the developer to further switch context and start looking at the other possibly relevant files.

TABLE III. CONTEXT SWITCHING COMPARISON: THE NUMBER OF MOUSE CLICKS REQUIRED BY THE DEVELOPER TO SEE THE ACTUAL CODE EXAMPLE.

Data Point	Code Repository Tools				
	Google Search	***Krugle***	***Koders***	***Stack Overflow***	***Example Overflow***
Dynamic Dimension	7	-	-	1	0
Hover	1	3	1	1	0
Position	5	-	-	7	0
Rounded Corners	3	-	5	5	0
Draggable	1	-	5	3	0
Dropable	1	-	5	2	0
Autocomplete	3	-	-	1	0
Accordion	1	-	23	6	0
Date Picker	1	-	5	1	0
Image Scale	3	-	-	-	0
Avg. Mouse Clicks	2.6	3	7.3333	3	0

VII. DISCUSSION

Searching for code examples is possible using Stack Overflow directly. However Example Overflow is better optimized for this use case, as our preliminary evaluation suggests. Although our approach uses data taken from Stack Overflow, we show different results, since we analyze the data differently and we use our own example-targeted search formula as shown in (1).

VIII. LIMITATIONS AND THREATS TO VALIDITY

Being part of the Example Embedding Ecosystem is a double-edged sword, because Example Overflow is not only enabling the ecosystem, but is also *being enabled* by it. Without proper *training*, the developer would not be able to critically evaluate the various examples, browse them and merge them. Without proper *practices*, systems which are developed using examples extensively may end up as *Frankenstein code* [2], and bugs may find their way in because the examples used were not properly tested.

Moreover, it is still unknown if crowd sourced software development would be able to scale well, as currently, Stack Overflow has only relatively small code snippets.

The preliminary evaluation provided above is limited; we examined only a small subset of programming tasks, with mostly popular tasks. The queries were phrased by one of the authors, who also determined the relevance of the results. Further evaluation is required, both qualitative and quantitative, involving professional developers.

IX. Future Work

Example Overflow was designed as a generic system to support additional domains and with a possibility to support other social media websites in addition to Stack Overflow.

Our future work will focus on integrating our tool into the IDE (Similarly to Blueprint [5] and Strathcona [11]) to further minimize the developer's context switching. Moreover, this will allow to run the example code in a sandbox mode before deciding whether it's suitable or not, and auto embedding the example code into the existing code (similarly to refactoring). This will also allow to auto suggest search queries by using the developer's structural context. By accomplishing these steps, the usage of examples will become an integral part of the software development cycle.

We plan to conduct a user study in which we observe professional developers as they are using our tool and learn how it is actually being used by the community. We hope to be able to characterize a set of best practices involved with example centric development. We also plan to data mine the system logs in order to reveal interesting patterns and fine tune the scoring formula (1).

Acknowledgment

We would like to thank Mati Shomrat for his valuable comments and helpful suggestions.

References

[1] D. Bajic and K. Lyons. Leveraging social media to gather user feedback for software development. In *Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering*, Web2SE '11, pages 1–6, New York, NY, USA, 2011. ACM.

[2] O. Barzilay. Example embedding. In *Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software*, ONWARD '11, pages 137–144, New York, NY, USA, 2011. ACM.

[3] B. Bibeault and Y. Katz. *jQuery in Action*. Manning Publications, 2nd edition, 2010.

[4] G. Bougie, J. Starke, M.-A. Storey, and D. M. German. Towards understanding twitter use in software engineering: preliminary findings, ongoing challenges and future questions. In *Proceeding of the 2nd international workshop on Web 2.0 for software engineering*, Web2SE '11, pages 31–36, New York, NY, USA, 2011. ACM.

[5] J. Brandt, M. Dontcheva, M. Weskamp, and S. R. Klemmer. Example-centric programming: integrating web search into the development environment. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 513–522, New York, NY, USA, 2010. ACM.

[6] M. Grechanik, C. Fu, Q. Xie, C. McMillan, D. Poshyvanyk, and C. M. Cumby. A search engine for finding highly relevant applications. In *ICSE (1)*, pages 475–484, 2010.

[7] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 53–60, New York, NY, USA, 2009. ACM.

[8] E. Hatcher, O. Gospodnetic, and M. McCandless. *Lucene in Action*. Manning, 2nd revised edition. edition, 8 2010.

[9] T. Hattori. Wikigramming: a wiki-based training environment for programming. In *Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering*, Web2SE '11, pages 7–12, New York, NY, USA, 2011. ACM.

[10] D. R. Hofstadter. Analogy as the core of cognition. In D. Gentner, K. J. Holyoak, and B. N. Kokinov, editors, *The Analogical Mind: Perspectives from Cognitive Science*, pages 499–538. MIT Press/Bradford Books, 2001.

[11] R. Holmes and G. C. Murphy. Using structural context to recommend source code examples. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 117–125. ACM, 2005.

[12] O. Hummel, W. Janjic, and C. Atkinson. Code conjurer: Pulling reusable software out of thin air. *IEEE Software*, 25(5):45–52, 2008.

[13] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen. A study of the difficulties of novice programmers. *SIGCSE Bull.*, 37:14–18, 2005.

[14] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann. Design lessons from the fastest Q&A site in the west. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 2857–2866, New York, USA, 2011. ACM.

[15] C. McMillan, D. Poshyvanyk, and M. Grechanik. Recommending source code examples via api call usages and documentation. In *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, RSSE '10, pages 21–25, New York, NY, USA, 2010. ACM.

[16] L. Novik. Analogical transfer, problem similarity, and expertise. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14:510–520, 1988.

[17] C. Parnin and C. Treude. Measuring api documentation on the web. In *Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering*, Web2SE '11, pages 25–30, USA, 2011. ACM.

[18] J. Stylos and B. Myers. Mica: A web-search tool for finding api components and examples. In *Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006.* pages 195 –202, 2006.

[19] A. Sureka, A. Goyal, and A. Rastogi. Using social network analysis for mining collaboration data in a defect tracking system for risk and vulnerability analysis. In *Proceedings of the 4th India Software Engineering Conference*, ISEC '11, pages 195–204, New York, NY, USA, 2011. ACM.

[20] J. Surowiecki. *The Wisdom of Crowds*. Anchor, 2005.

[21] S. Thummalapenta and T. Xie. Parseweb: a programmer assistant for reusing open source code on the web. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, ASE '07, pages 204–213, New York, USA, 2007. ACM.

[22] C. Treude, O. Barzilay, and M.-A. Storey. How do programmers ask and answer questions on the web? (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 804–807, New York, NY, USA, 2011. ACM.

[23] L. von Ahn. Human computation. In *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, pages 418 –419, july 2009.

[24] M. Vukovic. Crowdsourcing for enterprises. In *Services - I, 2009 World Conference on*, pages 686 –692, july 2009.

[25] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Trans. Inf. Syst.*, 26:13:1–13:37, June 2008.