Code duplication on stack overflow

Sebastian BALTES

Christoph TREUDE
Singapore Management University, ctreude@smu.edu.sg

Code Duplication on Stack Overflow

Sebastian Baltes
sebastian.baltes@adelaide.edu.au
The University of Adelaide, Australia

Christoph Treude
christoph.treude@adelaide.edu.au
The University of Adelaide, Australia

ABSTRACT

Despite the unarguable importance of Stack Overflow (SO) for the daily work of many software developers and despite existing knowledge about the impact of code duplication on software maintainability, the prevalence and implications of code clones on SO have not yet received the attention they deserve. In this paper, we motivate why studies on code duplication within SO are needed and how existing studies on code reuse differ from this new research direction. We present similarities and differences between code clones in general and code clones on SO and point to open questions that need to be addressed to be able to make data-informed decisions about how to properly handle clones on this important platform. We present results from a first preliminary investigation, indicating that clones on SO are common and diverse. We further point to specific challenges, including incentives for users to clone successful answers and difficulties with bulk edits on the platform, and conclude with possible directions for future work.

CCS CONCEPTS

• **Software and its engineering → Maintaining software**;

KEYWORDS

code duplication, code clones, software maintenance, software evolution, software licenses, stack overflow

ACM Reference Format:
Sebastian Baltes and Christoph Treude. 2020. Code Duplication on Stack Overflow. In *New Ideas and Emerging Results (ICSE-NIER'20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3377816.3381744

1 INTRODUCTION

Code clones have been extensively studied in the software engineering research community. Juergens et al. found that inconsistent code clones can be a major problem during the development and maintenance of software projects unless *"special care is taken to find and track existing clones and their evolution"* [1]. Stack Overflow (SO) threads, often containing code snippets together with explanations [2], serve as an important crowd-sourced software documentation resource [3, 4]. Despite the fact that code clones on SO can suffer from similar issues as code clones in software projects, it is only recently that researchers started investigating them. Studies have shown that developers utilise code snippets from SO in their software projects, regardless of maintainability, security, and licensing implications [5–14]. The main focus of that previous work was, however, to study how and why developers (re-)use SO code snippets outside of the question-and-answer platform. While researchers worked on identifying duplicate questions [15–17], their main goal was to replace or support the manual moderator process for marking duplicate questions rather than supporting the maintenance and evolution of code on SO. Considering the importance that SO has today for the daily work of many software developers worldwide and the fact that in many posts, non-trivial code snippets are collected and maintained, it is surprising that SO does not have proper features for code versioning and bug tracking. Text and code are versioned together as Markdown content [18], making it hard to identify changes to the code snippets in the provided revision view. Furthermore, there is no language-specific syntax highlighting or error checking in SO's online Markdown editor, leading to many snippets that are not parseable, compilable, or even runnable [2]. Finally, there is no way to report bugs in SO code snippets other than posting a comment or an alternative answer. Despite the above-mentioned challenges, code *is* maintained and *does* evolve on SO [18].

The purpose of this article is to point the research community to open questions regarding code clones on SO and to motivate how research in that area could inform significant improvements for the platform. We present a preliminary analysis of code clones within SO and point to directions for future work. As outlined above, the external usage of code snippets from SO has been studied in-depth. Therefore, our main focus is on code clones *within* the platform (see Section 3 for our specific research questions). While we are in an early stage of this research project, we already shared preliminary results with the SO community and started a discussion about how to handle code clones on the platform. The corresponding thread[1] attracted over 1,000 views and got upvoted to a score of 45. We will use the community's feedback to guide our next steps. Our vision is that a thorough study of code clones on SO together with the ongoing discussion in the SO community will lead to revised recommendations for authors and improved tool support for handling those clones.

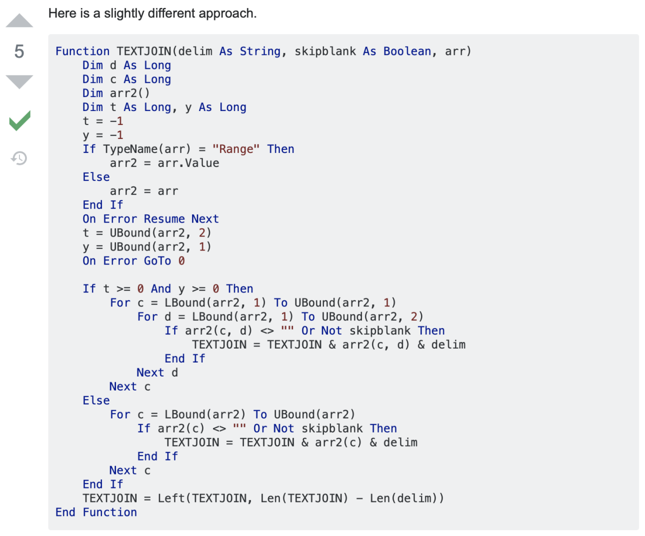
2 MOTIVATING EXAMPLES

In this paper, we want to discuss two use cases of duplicated code on SO that have not been adequately targeted by previous research. The first case covers content that was originally posted on SO, without any indication of an external source. For such snippets, the (visible) evolution mainly takes place on the SO platform, but nevertheless maintenance issues may arise. The second case is content that was

ICSE-NIER'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7126-1/20/05…$15.00
https://doi.org/10.1145/3377816.3381744

[1]https://meta.stackoverflow.com/q/375761


5

Here is a slightly different approach.

```
Function TEXTJOIN(delim As String, skipblank As Boolean, arr)
    Dim d As Long
    Dim c As Long
    Dim arr2()
    Dim t As Long, y As Long
    t = -1
    y = -1
    If TypeName(arr) = "Range" Then
        arr2 = arr.Value
    Else
        arr2 = arr
    End If
    On Error Resume Next
    t = UBound(arr2, 2)
    y = UBound(arr2, 1)
    On Error GoTo 0

    If t >= 0 And y >= 0 Then
        For c = LBound(arr2, 1) To UBound(arr2, 1)
            For d = LBound(arr2, 1) To UBound(arr2, 2)
                If arr2(c, d) <> "" Or Not skipblank Then
                    TEXTJOIN = TEXTJOIN & arr2(c, d) & delim
                End If
            Next d
        Next c
    Else
        For c = LBound(arr2) To UBound(arr2)
            If arr2(c) <> "" Or Not skipblank Then
                TEXTJOIN = TEXTJOIN & arr2(c) & delim
            End If
        Next c
    End If
    TEXTJOIN = Left(TEXTJOIN, Len(TEXTJOIN) - Len(delim))
End Function
```

Figure 1: First usage of a VBA code snippet on 16 September 2016 in answer 39532855, the most recent usage by the same user was on 27 August 2018 in answer 52040136. Within that timeframe, 31 copies of the same snippet were posted on SO, most of them by the same user. Consistently changing the snippet would require a manual update of all those copies.

copied from external sources into SO posts. Such clones suffer from the same maintenance issues as the clones mentioned above, but the additional external availability makes the evolution even more complex and may further introduce licensing issues.

2.1 Case 1: Original code snippets being reused in multiple threads

Figure 1 shows the first usage of a VBA (Visual Basic for Applications) code snippet, of which, within a time span of two years, 31 exact copies were posted on SO. Except for three questions, the snippet was exclusively used in different answers posted by the same user. None of those copies reference each other, meaning that an update in one copy would stay unnoticed in the other threads. The original author of the snippet managed to attract an accumulated number of over 25,000 views and an accumulated score of 65. Interestingly, it is not the initial answer that attracted the most views and the highest score. When we proposed to link more recent posts to the first occurrence, the author rejected our edit with a generic reply.[2] This leads to our first observation related to code duplication on SO:

> SO users may utilise code clones to accumulate views and upvotes. At the same time, they can reject edit proposals referencing the clones.

Note that we referred to exact copies in this example. It is likely that there exist further type-2 clones (e.g., with renamed identifiers) or type-3 clones (e.g., with added or removed statements) of the snippet. Some of those clones may contain fixes not yet propagated to the other clones.

[2]https://stackoverflow.com/review/suggested-edits/21495979

2.2 Case 2: Externally available snippets being reused in multiple threads

Not all snippets on SO are originally posted there, many are copied to and from the platform [6, 11, 14]. While this observation is not new, many existing studies focused on the implications of reusing content from SO, but not so much on the licensing and maintainability implications for the platform itself.

Considering SO's role in the software documentation landscape, it is not surprising that content from reference documentation resources is being reused on SO. Even in cases where a license-compatible usage of content would be straightforward, SO authors fail to adhere to license requirements. For example, a Java snippet about server certificate verification from an official Android tutorial[3] has been copied (at least) 14 times into SO. This happened in a timespan of over four years. Google licensed this tutorial—and thus the snippet—under CC BY 2.5. This license is compatible with SO's CC BY-SA license but requires attribution, which SO authors do not always provide. Besides this licensing aspect, in the above example, there is unarguably one authoritative source for the snippet where bug fixes or updates would be posted. Therefore, our next observation is:

> SO users copy code snippets from reference documentation into SO posts. Besides licensing and copyright implications, it is questionable whether this behaviour contributes to the sustainability of the platform, because users may reuse outdated information in case the authoritative source gets modified. Even if one of the clones gets updated, the changes are not automatically propagated within the network of clones on SO.

There is another angle to this case: We added the missing attribution in two posts containing the above-mentioned snippet from the official Android documentation. While those edits were accepted, there is a rate limited for such bulk edits.[4] Hence, we observe that:

> Due to SO's rate limiting, missing attribution cannot be easily added to posts. Tools that researchers build can currently only very slowly propagate proposed changes to affected posts.

3 RESEARCH QUESTIONS

Motivated by the two cases outlined above, we argue that the following research questions need to be addressed to be able to make an informed decision on how to handle code clones on SO. The goal of this research is to provide data-informed actionable recommendations which the community and SO's internal team can use to update their user guides, but also to build tool support for managing and maintaining code clones on the platform.

RQ1: How frequently are code snippets copied between SO posts?
RQ2: What types of clones exist and how are they related?
RQ3: What are typical external sources of code snippets on SO and which licensing and maintainability implications are associated with those sources?

The first research question helps us to assess how common the outlined problem is, the second to deepen our understanding of the

[3]https://developer.android.com/training/articles/security-ssl.html
[4]https://meta.stackexchange.com/a/281202

different use cases of clones on SO, and the last to understand the implications beyond the platform.

4 PRELIMINARY ANALYSIS

We conducted a preliminary analysis focusing on exact code clones on SO. While this approach has the advantage of not being limited to a certain programming language, extending the analysis to also cover other types of clones will be a logical next step.

4.1 Data Retrieval

To detect code clones on SO, we utilised the BigQuery version of *SOTorrent* [18]. First, we selected all code blocks from the most recent post versions and normalised the contained whitespace characters. To this end, we: (1) replaced sequences of new lines with a single new line character, (2) removed new lines at the end of the last line, and (3) removed lines only containing brackets (`()[]{}`). Using this normalised content, we calculated the normalised line count of those code blocks (NLOC). To derive fingerprints of the snippets, we only considered alphanumeric characters and applied BigQuery's `FARM_FINGERPRINT` function to the normalised code block contents. This yielded 43,942,960 distinct fingerprints (i.e., normalised code blocks). We then used this fingerprint to determine the posts which contain a certain snippet, aggregating that information per thread. To select cloned code blocks, we first selected the ones present in at least two different threads, yielding 909,323 distinct fingerprints. This provides a first estimate for **RQ1**: 2.1% of all distinct code blocks have a copy in another thread. To select only non-trivial code snippets, we first used a threshold of six normalised lines of code, as proposed by Bellon et al. [19]. We ranked the remaining 215,746 code snippets according to the number of threads they were found in and according to their normalised length. Then, we qualitatively analysed the first 50 snippets in that list. Since we considered 25 of those snippets to be either too trivial or non-code, we decided to adjust the threshold for the normalised line count to 20. The stricter filtering led to a second sample with 46,818 code snippets. Those snippets had an average length of 42.6 normalized lines ($SD = 37.7$, $Mdn = 30$, $IQR = 22$) and were present in 2.3 different threads ($SD = 1.1$, $Mdn = 2$, $IQR = 0$); 13.4% of the snippets were present in more than two threads (see Figure 2). We provide the retrieval scripts and the coding for both samples (≥ 6 NLOC and ≥ 20 NLOC) on Zenodo.[5]

4.2 Qualitative Analysis

To address **RQ2** and **RQ3**, we first ranked the code snippets according to their thread count and length and then qualitatively analysed the first 50 snippets according to that ranking using a web tool[6] we specifically designed for that purpose. The tool allows to focus on a single snippet in a dedicated view, showing the snippet, its fingerprint, the posts containing the snippet sorted by their creation date, other posts linked from those posts, and linked external sources (see Figure 2). The latter information helped us to identify if and from where a snippet may have been copied into SO. While we still categorised ten snippets as configuration files, 29 snippets were non-trivial source code snippets (mainly Java and VB/VBA). Other categories included XML GUI definitions for Android, JSON/XML examples, and HTML files. Except for two cases, we were able to identify the (or at least a) source of the snippet by following links in the posts and searching for parts of the snippets online. Only in four cases, we considered the snippets to be originally from SO. The main external sources were a website providing Android tutorials (*androidhive.info*, ten snippets) and the official Android documentation (*developer.android.com*, four snippets). We identified possible licensing conflicts in 31 cases, either because the website did not provide a license or because the content was distributed under a restrictive license or restrictive terms of use. In the following, we describe the two main external sources in more detail.

The independent Android website *androidhive* has rather restrictive terms of use. Nevertheless, only few posts attribute this source (3 out of 45 posts in the example shown in Figure 2). It is unclear whether the snippet has actually been copied from this external source since the creation of the posts on *androidhive* and SO were both around April/May 2012. If the 45 snippets were copied into SO, their usage would be problematic. In fact, we identified four more variants of that same code snippet among the 50 snippets we analysed. If SO is the original source, the usage on *androidhive* does not adhere to SO's CC BY-SA license [6]. The snippets copied from the *official Android documentation* are licensed under CC BY 2.5. This license allows usage under SO's CC BY-SA license, but only when attributing the original source. However, users often did not add a link to the Android documentation to their posts. Thus, also those usages could lead to *licensing issues*.

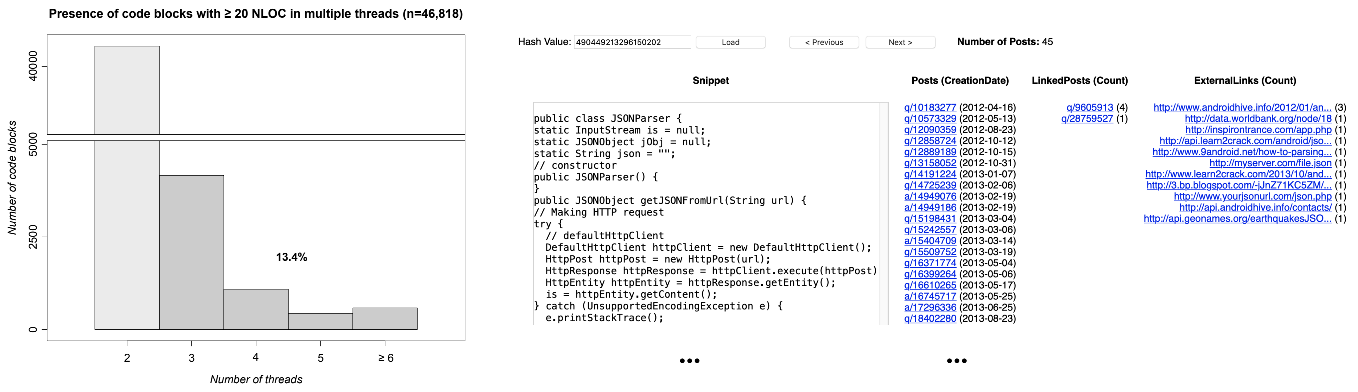
Leaving the licensing implications aside, code clones within SO are also problematic for the platform's *maintainability* and *usability*. Code duplicates could, for example, indicate that different threads solved a similar problem. If there is no link between the threads, information is spread over the platform and hard to capture for readers. Another example is SO's recommendation to *"always quote the most relevant part of an important link, in case the target site is unreachable or goes permanently offline"*.[7] While it makes sense to quote important aspects of external sources, it can be questioned whether it is reasonable to maintain several independent copies of external code snippets on SO. Assuming that a snippet in the reference documentation is updated, all copies on SO would require a manual update as well.

4.3 Community Involvement

To involve the community and discuss how to best approach those licensing, maintainability, and usability issues, we created a post on SO Meta. We outlined the two cases presented in Sections 2.1 and 2.2 using examples from our preliminary analysis and asked the community: *How to handle code clones on Stack Overflow?*

One preliminary observation from the discussion is that the community seems to be in favour of adding missing attribution to SO posts where it is missing. This would enable tool support for automatically checking external sources for updated versions of snippets, but only solve the licensing issue for snippets licensed under a rather permissive license. Regarding approaches to handle clones within SO, there is no clear opinion yet. Besides continuing to work on the research questions presented in Section 3, we will

[5]https://doi.org/10.5281/zenodo.1474222 and https://doi.org/10.5281/zenodo.3596367
[6]https://doi.org/10.5281/zenodo.1474207
[7]https://stackoverflow.com/help/how-to-answer



Hash Value: 490449213296150202

Number of Posts: 45

Snippet

```
public class JSONParser {
static InputStream is = null;
static JSONObject jObj = null;
static String json = "";
// constructor
public JSONParser() {
}
public JSONObject getJSONFromUrl(String url) {
// Making HTTP request
try {
  // defaultHttpClient
  DefaultHttpClient httpClient = new DefaultHttpClient();
  HttpPost httpPost = new HttpPost(url);
  HttpResponse httpResponse = httpClient.execute(httpPost)
  HttpEntity httpEntity = httpResponse.getEntity();
  is = httpEntity.getContent();
} catch (UnsupportedEncodingException e) {
  e.printStackTrace();
```

Posts (CreationDate)	LinkedPosts (Count)	ExternalLinks (Count)
q/10183277 (2012-04-16)	q/9605913 (4)	http://www.androidhive.info/2012/01/an... (3)
q/10573329 (2012-05-13)	q/28759527 (1)	http://data.worldbank.org/node/18 (1)
q/12090359 (2012-08-23)		http://inspirontrance.com/app.php (1)
q/12858724 (2012-10-12)		http://api.learn2crack.com/android/jso... (1)
q/12889189 (2012-10-15)		http://www.9android.net/how-to-parsing... (1)
q/13158052 (2012-10-31)		http://myserver.com/file.json (1)
q/14191224 (2013-01-07)		http://www.learn2crack.com/2013/10/and... (1)
q/14725239 (2013-02-06)		http://3.bp.blogspot.com/-jJnZ71KC5ZM/... (1)
a/14949076 (2013-02-19)		http://www.yourjsonurl.com/json.php (1)
a/14949186 (2013-02-19)		http://api.androidhive.info/contacts/ (1)
q/15198431 (2013-03-04)		http://api.geonames.org/earthquakesJSO... (1)
q/15242557 (2013-03-06)		
a/15404709 (2013-03-14)		
q/15509752 (2013-03-19)		
q/16371774 (2013-05-04)		
q/16399264 (2013-05-06)		
q/16610265 (2013-05-17)		
a/16745717 (2013-05-25)		
a/17296336 (2013-06-25)		
q/18402280 (2013-08-23)		

•••   •••

Figure 2: *Left:* Copies of non-trivial code blocks (≥ 20 NLOC) in multiple SO threads. *Right:* Snippet view of `so-clones` tool showing a code snippet that has likely been copied from the website *androidhive* into SO.

update the discussion and finally implement the approach that the community prefers. Possible directions include to automatically propose post edits adding missing attribution, to automatically link to the first occurrence in a set of duplicates, or to mark threads as related based on the similarity of the code blocks they contain. The existing Guttenberg bot monitors newly posted questions and answers and automatically reports plagiarism, mainly based on String similarity between posts. The bot does, however, compare whole posts without isolating code snippets first. When we ran our study, the bot was already active. Nevertheless, we still found a considerable amount of duplicated code on the platform. We have shared our results with the Guttenberg team.

5 CONCLUSION

With this paper, we want to point to the fact that code clones on SO, similar to clones in regular software projects, affect the maintainability of posts and can lead to licensing issues. However, we also point to differences such as the fact that SO users may be encouraged to clone successful answers to achieve a higher reputation and that snippets are difficult to modify through bulk edits. These differences have practical implications and might suggest new features for SO, such as allowing bulk edits for adding attribution information or improving detection of overly zealous self-plagiarism. We also mentioned that SO's current recommendation for handling content from external resources may not be suitable for code taken from reference documentation, because the authoritative source should be the reference documentation and not SO. Moreover, when clones on SO are not updated, users consulting SO threads instead of the reference documentation may be prone to using outdated or erroneous information.

Our preliminary results suggest that code duplication on SO is relatively common (**RQ1**). Despite our limitation on exact clones, we found that 2.1% of all unique code blocks were used in more than one thread. Moreover, when we focused on unique code blocks with ≥ 20 NLOC, we still found 47k of them being used in more than one thread. Our analysis also revealed that a wide variety of snippets is being cloned (**RQ2**). We noticed cases where the same user copied the same snippet into several answers in different threads. As motivated in Section 2, future work should investigate this behaviour. Further, the external sources of code snippets on SO (**RQ3**) range from random blog posts to official reference documentation pages (see our Android example). Identifying typical external sources and developing ways to keep the content on SO up-to-date is a promising direction for future work. This research could even be extended to include other online forums or code hosting platforms.

REFERENCES

[1] E. Juergens, F. Deissenboeck, B. Hummel, and S. Wagner, "Do Code Clones Matter?" in *ICSE 2009*.
[2] D. Yang, A. Hussain, and C. V. Lopes, "From Query to Usable Code: An Analysis of Stack Overflow Code Snippets," in *MSR 2016*.
[3] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, "Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow," *Georgia Institute of Technology, Technical Report*, 2012.
[4] C. Treude, O. Barzilay, and M.-A. D. Storey, "How do programmers ask and answer questions on the web?" in *ICSE 2011*.
[5] S. Baltes, R. Kiefer, and S. Diehl, "Attribution required: Stack overflow code snippets in GitHub projects," in *ICSE 2017 Companion Volume*.
[6] S. Baltes and S. Diehl, "Usage and Attribution of Stack Overflow Code Snippets in GitHub Projects," *Empirical Software Engineering*, vol. 24, no. 3, 2019.
[7] L. An, O. Mlouki, F. Khomh, and G. Antoniol, "Stack Overflow: A Code Laundering Platform?" in *SANER 2017*.
[8] D. Yang, P. Martins, V. Saini, and C. V. Lopes, "Stack Overflow in Github: Any Snippets There?" in *MSR 2017*.
[9] M. Gharehyazie, B. Ray, and V. Filkov, "Some From Here, Some From There: Cross-Project Code Reuse in GitHub," in *MSR 2017*.
[10] R. Abdalkareem, E. Shihab, and J. Rilling, "On code reuse from StackOverflow: An exploratory study on Android apps," *IST Journal*, vol. 88, 2017.
[11] X. Xia, L. Bao, D. Lo, P. S. Kochhar, A. E. Hassan, and Z. Xing, "What do developers search for on the web?" *Empirical Software Engineering*, vol. 22, no. 6, 2017.
[12] F. Fischer, K. Boettinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security," in *S&P 2017*.
[13] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, "You Get Where You're Looking For: The Impact Of Information Sources on Code Security," in *S&P 2016*.
[14] Y. Wu, S. Wang, C.-P. Bezemer, and K. Inoue, "How do developers utilize source code from stack overflow?" *Empirical Software Engineering*, vol. 34, no. 2, 2018.
[15] Y. Zhang, D. Lo, X. Xia, and J. Sun, "Multi-Factor Duplicate Question Detection in Stack Overflow," *Journal of Computer Science and Tech.*, vol. 30, no. 5, 2015.
[16] M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider, "Mining duplicate questions in stack overflow," in *MSR 2016*.
[17] W. E. Zhang, Q. Z. Sheng, J. H. Lau, and E. Abebe, "Detecting Duplicate Posts in Programming QA Communities via Latent Semantics and Association Rules," in *WWW 2017*.
[18] S. Baltes, L. Dumani, C. Treude, and S. Diehl, "SOTorrent: Reconstructing and Analyzing the Evolution Stack Overflow Posts," in *MSR 2018*.
[19] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo, "Comparison and evaluation of clone detection tools," *IEEE Transactions on Software Engineering*, vol. 33, no. 9, 2007.