2015 IEEE 23rd International Conference on Program Comprehension

Synonym Suggestion for Tags on Stack Overflow

Stefanie Beyer
Software Engineering Research Group
University of Klagenfurt
Klagenfurt, Austria
Email: stefanie.beyer@aau.at

Martin Pinzger
Software Engineering Research Group
University of Klagenfurt
Klagenfurt, Austria
Email: martin.pinzger@aau.at

***Abstract*—The amount of diverse tags used to classify posts on Stack Overflow increased in the last years to more than 38,000 tags. Many of these tags have the same or similar meaning. Stack Overflow provides an approach to reduce the amount of tags by allowing privileged users to manually create synonyms. However, currently exist only 2,765 synonym-pairs on Stack Overflow that is quite low compared to the total number of tags.**

To comprehend how synonym-pairs are built, we manually analyzed the tags and how the synonyms could be created automatically. Based on our findings, we then present TSST, a tag synonym suggestion tool, that outputs a ranked list of possible synonyms for each input tag.

We first evaluated TSST with the 2,765 approved synonym-pairs of Stack Overflow. For 88.4% of the tags TSST finds the correct synonyms, for 72.2% the correct synonym is within the top 10 suggestions. In addition, we applied TSST to 10 randomly selected Android related tags and evaluated the suggested synonyms with 20 Android app developers in an online survey. Overall, in 80% of their ratings, developers found an adequate synonym suggested by TSST.

I. Introduction

Tags are part of social bookmarking, a service of Web 2.0 to classify and label data in an informal way [1], [2]. Tagging is also used on Q&A-sites, such as Stack Overflow, to categorize questions. Several recent research approaches have focussed on the extraction of topics and trends on Stack Overflow, and tags seem to be a good point to start from. However, they also found that tags are often too fine grained or too inconsistent for their purposes [3].

In September 2014, there were more more than 38,000 different tags on Stack Overflow. There is an approach of Stack Overflow to reduce the large number of tags by suggesting synonym pairs, consisting of tags that have been created by privileged users. These synonym pairs are manually suggested and evaluated, and if they are accepted, they may be used. At the time of September 2014, there were 2,765 synonym-pairs on Stack Overflow consisting of 4,593 different tags. Understanding how the synonyms are built and how they may be automated could improve studies using tags for a categorization of posts or finding topics and trends on Stack Overflow.

In this paper, we first investigate strategies how synonym-pairs of Stack Overflow are built. Then, we use these findings to develop a synonym suggestion tool called TSST that implements theses strategies. For a given input tag, TSST outputs a ranked list of suggested synonyms. With this research, we address the following three research questions:

– RQ1: How are the tag synonyms of Stack Overflow built?
– RQ2: How many of the existing tag synonyms on Stack Overflow can be built with each strategy?
– RQ3: How accurate is TSST in suggesting synonyms?

Regarding RQ1, we manually analyzed the set of synonym-pairs on Stack Overflow and discovered 9 different strategies, how synonyms are created. Based on these strategies, we developed TSST that we first evaluated with the set of synonym-pairs. Answering RQ2, we first analyzed the percentage of Stack Overflow synonym-pairs correctly created by each strategy. It turned out that Metaphone and Synonym-In-Word are the two most generic strategies to create synonyms. Furthermore, we found a significant overlap between several strategies.

For answering RQ3, we evaluated TSST with the Stack Overflow synonym-pairs and, in addition, with an online survey. Regarding the evaluation with the synonym-pairs, we investigated if the correct synonym is found within the top 3, top 5, top 10, or top 15 synonyms suggested by TSST. We found that 88.4% of the synonyms are suggested correctly, out of them 67.9% are within the top 5 suggested synonyms and for 45.9% the first suggestion was the correct one.

Concerning the online survey, we first applied TSST to 10 randomly selected tags related to Android specific posts on Stack Overflow, and then evaluated the suggestions with 20 Android app developers. Overall, in 80% of their ratings, developers found an adequate synonym suggested by TSST within the top 15 suggestions.

In this paper, we make the following contributions:

– A manual analysis of 9 strategies to systematically recreate synonyms.
– A study of how many synonym-pairs on Stack Overflow can be found using which strategy.
– TSST, a tag synonym suggestion approach and tool.
– An evaluation of TSST with the Stack Overflow synonym-pairs and 20 Android app developers.

The remainder of this paper is organized as follows. In Section II, we provide background information to the creation of tags and tag-synonyms on Stack Overflow. In Section III, we describe the analysis of the tags and strategies to find synonyms automatically. Furthermore, we present the answers

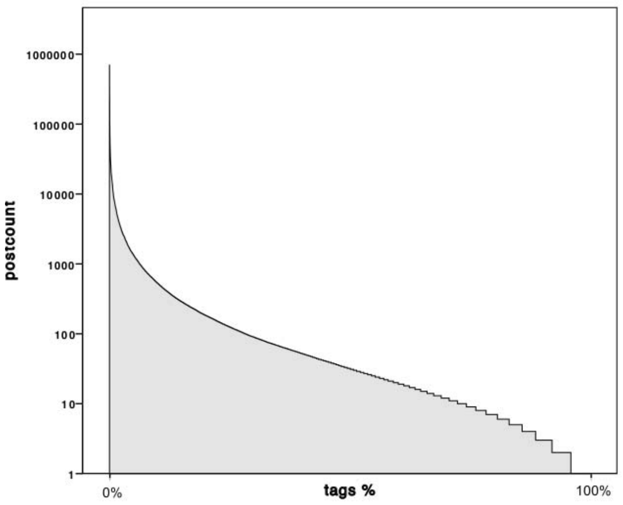

Fig. 1. Distribution of the usage of tags (postcount) on Stack Overflow (log-scale).

to the research questions RQ1 and RQ2. In Section IV, we introduce the tag synonym suggestion tool TSST. In Section V, we evaluate its accuracy and performance and answer research question RQ3. The applicability of the results, as well as their limitations and threats to validity are discussed in Section VI. Related work is presented in Section VII and we draw the conclusions and discuss future work in Section VIII.

II. Tags and Synonyms on Stack Overflow

In September 2014, there were 7,990,787 questions on Stack Overflow belonging to various challenges and problems of programming. To find relevant questions and answers easier, each post is labeled with one to five tags. Each questioner is allowed to tag her post, but only Stack Overflow users with a reputation of at least 1.500 have the privilege to create new tags. Users gain reputation, for instance, if a question or answer of the user is voted up, or an answer is marked 'accepted'. Users lose reputation, for instance, if a question or answer is voted down or if the user itself votes an answer down. The data dump from September 2014 contains 38,205 different tags. Among the most frequently used tags are `java`, `c#`, `javascript`, `php`, and `android`. The tag `java` is used more than 700,000 times, the tag `android` more than 560,000 times.

Having a look at the distribution of the usage of tags on Stack Overflow shown in Figure 1, we see that 25,74% of the tags are used less than 10 times and only 10.40% of the tags are used more than 500 times. The comparison of these numbers to the most frequently used tags, which are used more than 700,000 times, indicates that many tags have the same or similar meaning, are too specific, or too general. Another reason for this large number of different tags may be the fact that all these tags were created by users and the privilege needed for creating new tags was initially configured too low. This is indicated by a steady update of this limit over time from a reputation of 250, then to 500, and finally to 1,500.[1]

One measure taken by Stack Overflow to reduce the amount of new tags is to cull single-use tags, if they are older than 6 months and do not have a wiki.[2] Furthermore, Stack Overflow provides a feature to manually create synonyms for each tag. On Stack Overflow two tags are a *synonym-pair* if both tags have the same meaning, such as `jpeg` and `jpg` or one tag is a subset of the other tag, such as `encoding` and `character-encoding`.[3]

In September 2014, there were 2,765 synonym-pairs on Stack Overflow. These synonyms have been created manually by users of Stack Overflow. All users having a reputation $>= 2,500$ are allowed to suggest synonyms. These suggestions are rated by other users. If the score is $>= 5$, the suggestion is approved and the synonym may be used for tagging. If the score becomes $<= -2$ the synonym suggestion is declined and deleted.

Each synonym-pair consists of a *source tag* and a *target tag*. The target tag is more general than the source tag and it replaces internally all uses of the source tag. For instance, by searching questions tagged with a synonym, questions tagged with the target tag are displayed, or when a question is tagged with a synonym, the target tag is displayed when loading the question. For each tag there exists only one target tag. Target tags may have more than one source tag.

Tags are often used as additional information for the categorization of posts or for topic modeling [3], [4]. The knowledge about synonyms could improve studies and approaches by finding redundant tags and grouping them together. However, the amount of manually created synonym-pairs compared to the number of existing tags is low. This motivates our analysis of the synonym-pairs to find strategies how they are built with the goal to automate tag synonym suggestion.

III. Tag Synonym Analysis

We extracted the list of tags from the data dump of Stack Overflow, provided by Stack Exchange from September 2014. The list of synonyms is not available in the dump, therefore we extracted the list of synonym-pairs from the Stack Exchange data explorer.[4] We select only the tag synonyms that were created before September 2014.

The synonym-pairs may also be in a transitive relation, for example `rng` is the source tag of `random-number-generator` and `random-number-generator` is the source tag of `random`. Consequently, `random` should also be the target tag for `rng`. Analyzing the Stack Overflow tags, we found that tags are composed of 1 to 5 words that are separated

[1]http://blog.stackoverflow.com/2010/08/tag-folksonomy-and-tag-synonyms/

[2]http://meta.stackoverflow.com/questions/272094/do-not-automatically-expire-single-use-tags-on-stack-overflow

[3]http://meta.stackexchange.com/questions/70710/what-are-tag-synonyms-how-do-they-work

[4]http://data.stackexchange.com

with a '-' or '.'. In the remainder of the paper, we refer to these parts as *pots* meaning part of a tag.

To get more insights on how the synonym-pairs are composed, we manually analyzed the 2,765 synonym-pairs of Stack Overflow and found 9 strategies. In the following, we discuss the strategies and present the answer to research question RQ1:

RQ1 - How are the tag synonyms of Stack Overflow built?

As an answer to the question, we found the following 9 strategies:

– Stemming
– Synonym-In-Word
– Synonym-In-Tag
– Similarity
– Acronym
– DotSharpMinusPlus
– Abbreviation/Synonym
– Metaphone
– Numbers

In the following, we describe each strategy in detail and explain our approach to automate each one.

Stemming: Synonym-pairs that are built with the *Stemming* strategy often consist of the singular and plural noun of the same word. Tags that stem from the same word are also often grouped to synonym-pairs. We automate this strategy with the Porter Stemmer [5], provided by Apache Lucene,[5] that cuts the ending of the words and matches the tags by the stems of the words. Examples for such synonym-pairs are: `algorithm` and `algorithms` or `clustered-indexing` and `clustered-index`.

Synonym-In-Word: There are two possibilities, how tags are built with the *Synonym-In-Word* strategy. The first one matches two tags if one tag is completely contained by the other tag. The second possibility to match tags is that one *pot* matches the beginning or end of another tag that does not consist of *pots*. To automate this strategy, we first stem the tag and look for other tags that start or end with this tag. If the tag has *pots*, we stem each *pot* and search again for tags that start or end with this part. Synonym-pairs that are built with this strategy are, for instance, `threading` and `multithreading` or `play-mvc` and `playframework`. We considered the splitting of composed words and found libraries that split words by camel-case. However, there are no tags consisting of a capitalized letter and therefore we left the splitting of words into words for future work.

Synonym-In-Tag: Tags are composed with the *Synonym-In-Tag* strategy, if they have at least one *pot* in common. To automate this strategy, we split the tag into *pots*, stem the *pots* and built synonym-pairs that have at least one stemmed *pot* in common. Examples for synonym-pairs built with this strategy are `android-sdk` and `android` or `nested-class` and `inner-classes`.

Similarity: The name of this strategy already reveals that tags with similar characters are matched to synonym pairs. This strategy is often used if there are misspellings or variant spellings for tags. We automate this strategy by using three kinds of string similarity metrics, namely the Jaccard-Index, the Levenshein-Distance and the NGram-Distance. The Jaccard-Index [6] calculates the number of characters in common divided through the number of different characters. If the Jaccard-Index is 1, the two tags consist of exactly the same characters. The order of the characters is not considered. The Levensthein-Distance [7] is calculated by counting the number of edit-operations that are required to change one tag into the other. Edit operations are, for instance, insertions, deletions, and substitutions. The NGram-Distance, based on Kondrak [8], computes the partial matches of substrings of size n. We set n to the values 2, 3, and 4. The implementation of the Levensthein-Distance and NGram-Distance is provided by Apache Lucene.[5] We decided to use all similarity metrics, since they differ in accuracy and implementation. The Jaccard-Index is less accurate than the Levensthein-Distance and the Levensthein-Distance is less accurate than the NGram-Distance. Based on experiments, we evaluated the best settings for the limits to match similar tags and set the limit for Jaccard-Index to 0.75, for the Levensthein-Distance to 0.7, and for the NGram-Distance to 0.6. Synonym-pairs that are found using this strategy are, for instance, `perfomance` and the correct spelled tag `performance`, or `tchart` and `teechart`.

Acronym: The synonym-pairs that are built with *Acronym* consist of an abbreviation that is composed, in the simple case, of the first characters of some concatenated words. To automate the creation of an acronym, we take the first character of each *pot* of a tag and compose them to an acronym. There are special cases, when we did take the first character of each part. If a *pot* is `to`, we put a `2` instead of a `to`. The same goes for `cross` and `x`, `and` and `n`. Furthermore, we also compute complex synonyms, where all combinations of the first character, first and second character, first to third character of all *pots* are composed and matched to a tag that starts or ends with this abbreviation. Synonym-pairs that are created with this strategy are, for instance, `peer-to-peer` and `p2p` or `user-interface` and `ui`.

DotSharpMinusPlus: The strategy *DotSharpMinusPlus* replaces a character with another predefined character or the literal name of the character. The character `.` is replaced by `dot`, `#` is substituted by `sharp`, `-` is removed, and the sequence `++` is replaced by `pp`. The substitutions are also applied vice versa. To automate this strategy, we use the Java String API to substitute and remove the characters. Examples for synonym-pairs following this strategy are: `.net` and `dot-net`, `c#` and `csharp`, `for-xml` and `forxml` or `c++` and `cpp`.

Abbreviation/Synonym: Synonym-pairs that are built with this strategy are synonyms or abbreviations for which we could not find a schema or pattern how they are created. There are dictionaries and synonym-sites that provide a list of synonyms for download, such as thesaurus.[6] With these data the matching of synonyms could be automated. However, domain specific

[5]http://lucene.apache.org

[6]http://www.thesaurus.com

abbreviations, such as tdd for testdrivendevelopment or db for database are not covered. As a consequence, the implementation of the automation of finding abbreviation-synonyms is left for future work. Synonym-pairs that we approach to find with this strategy are discussion and conversation or text-message and sms.

Metaphone: If the pronunciation of tags is similar, they are composed into synonym-pairs with the *Metaphone* strategy. This strategy is used to find tags with the same meaning but with variant spelling, such as behaviour and behavior. Furthermore, it helps to match misspelled words to their correct spelled synonyms, such as heirarchie and hierarchie. To automate this strategy, we use the Metaphone algorithm [9], an improved version of the phonetic algorithm Soundex. Metaphone indexes the tags by their pronunciation and is provided by Apache Commons.[7] The length of the Metaphone code depends on the size of the tag, but it has a minimum of 2 and a maximum of 7 characters. Another synonym-pair that is built with *Metaphone* is, for instance, jpg and jpeg.

Numbers: The last strategy we found to create synonyms, is *Numbers*. Synonym-pairs are created with this strategy, when tags containing numbers match to other tags by either replacing this number with the number in literals or vice versa. Tags are also matched if they are equivalent if all numbers are removed. To automate this we check each tag that contains numbers twice. First, we programmatically replace all numbers of a tag with their literal number and search for matches. Second, we remove all numbers and check for tags that are equivalent, if the numbers are removed. Synonym-pairs that are created using this strategy are, for instance, 7zip and sevenzip or joomla-3.1 and joomla-3.0.

In the following, we present the evaluation of the found strategies and present the answer to the research question RQ2:

RQ2 - How many of the existing tag synonyms on Stack Overflow can be built with each strategy

To check if the strategies stated above cover all approved synonym-pairs of Stack Overflow, we checked for each synonym-pair programmatically the strategy used to create the synonym-pair. Strategies, such as *Stemming* and *Synonym-In-Word* often overlap with each other.

Figure 2 shows the percentage of synonym-pairs that can be created using each strategy. Using the strategy *Synonym-In-Tag*, 1,429 of the 2,765 synonym-pairs were recreated, that is 51.7%. The strategy *Abbreviation* covers 599 synonym-pairs (21.7%), followed by *Synonym-In-Word* with 1,484 (53.7%), and *Metaphone* covering 1,489 synonym-pairs (53.9%). The strategy *Similarity* covers 1,390 synonym-pairs (50.3%), *Stemming* covers 561 (20.3%) synonym-pairs, *DotSharpMinusPlus* 121 (4.4%), and *Numbers* 12 (0.4%).

[7]http://commons.apache.org/proper/commons-codec/

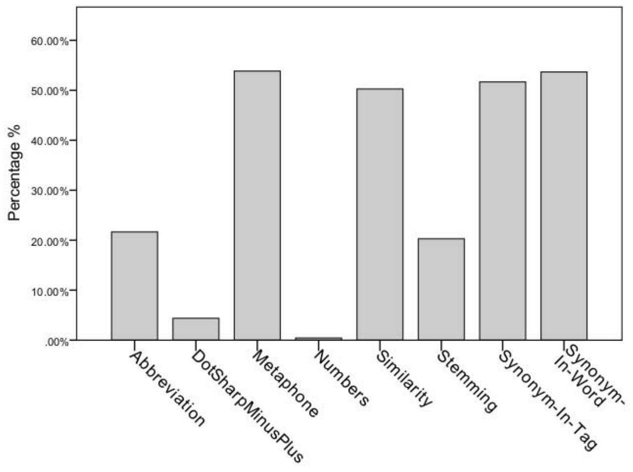

Fig. 2. Percentage of Stack Overflow synonym-pairs that can be created with each strategy.

Overall, with these strategies, we are able to create 2,445 of the set 2,765 synonym-pairs, that is 88.4% of all pairs from the reference set provided by Stack Overflow. Manually investigating the 320 synonym-pairs that could not be matched, we found 310 synonym-pairs falling into the strategy *Abbreviation/Synonym* that we have not automated, yet. Furthermore, for 10 synonym-pairs, we found variations of the strategy *Numbers*, such as y2k38 and year2038 that are not implemented, yet. We plan to address these two issues in our future work.

IV. TSST - Tag Synonym Suggestion Tool

To integrate our strategies, we developed TSST, a tag synonym suggestion tool. TSST takes one tag or a list of tags as input, as well as an integer value *maxSuggestions* for the maximum number of synonyms to suggest. Then, it generates synonym-candidates for each input tag and stores them into a table, consisting of a source tag, a target tag, a counter, and the used strategy. TSST outputs a ranked list of possible synonym candidates. Figure 3 shows the steps of the synonym finding process of TSST with the input tag class and a *maxSuggestions* of 5. The creation of the synonym-candidates is based on the strategies presented above.

We ordered the automation of the strategies from restrictive to general. At first, the strategy *Stemming* is applied, then *Numbers* and so on. *Metaphone* is the most general strategy. Synonym-pairs that belong to this category may also be in *Stemming*, *Numbers*, *DotSharpMinusPlus*, *Synonym-In-Word*, or *Similarity*. Therefore, we put it at the end. *Similarity* is more restrictive than *Metaphone*, since it considers the literals of the words, not only their pronunciation. Synonym-pairs built with *Acronym* are matched by the begin and end of other tags, therefore, we put it before *Similarity*. It occurs that synonym-pairs could be created with both, *Synonym-In-Tag* and *Synonym-In-Word*. However, *Synonym-In-Tag* is more specific than *Synonym-In-Word* and so we put it before

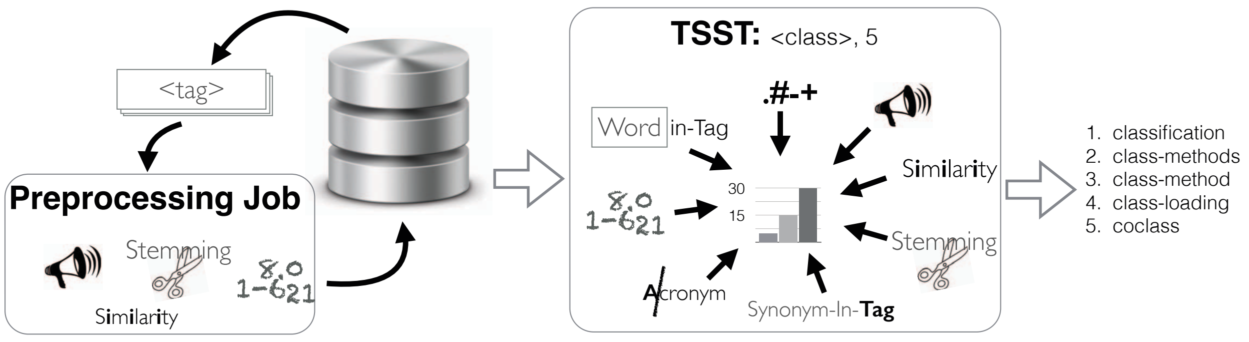

Fig. 3. Overview of TSST - Tag Synonym Suggestion Tool and the preprocessing job

Synonym-In-Word. We ordered the remaining strategies also from restrictive to general resulting in the order: Stemming - Numbers - DotSharpMinusPlus - Synonym-In-Tag - Synonym-In-Word - Acronym - Similarity - Metaphone.

The synonym-candidates output by each strategy are stored. If there is already an entry with this combination of tags, disregarding the source and target tag order, then the counter for this synonym-candidate is increased. To improve performance, we performed the calculations of the string similarity values, the computation of *pots* and tags without numbers and Metaphone codes for each combination of tags in a preprocessing job. Only string similarity values between tags of $>= 0.5$ are considered. The NGram-Distance is calculated for $n = 2, 3, 4$. Regarding Metaphone, we computed codes varying in length ranging from 2 to 7. The results of the preprocessing job are stored in a database and the job is run once for a given data set.

Ranking: Each synonym-candidate that is generated by a strategy has a counter c. Each time, a strategy creates a synonym-candidate that already exists, c is increased. To rank the synonym-candidates for each tag, we order them by the counter c in a descending order and output the suggestions in this order. Alternatively, we also considered to perform the ranking based on how often each strategy is used in the approved set of synonym pairs. We reject this idea, since our other approach to rank the synonyms performed better on the set of Stack Overflow synonym-pairs.

TSST is implemented in Java, using libraries of Lucene and Metaphone, provided by Apache, and a MySQL database for storing the results. A replication package consisting of the prototype implementation of TSST and a dump of the database are available on our website.[8]

V. Evaluation of TSST

In this section, we present the evaluation of TSST and answer to research question RQ3:

[8]http://serg.aau.at/bin/view/StefanieBeyer/TSST

RQ3 - How accurate is TSST in suggesting synonyms?

The accuracy of TSST is calculated in two phases. First, we evaluated TSST on the approved set of 2,765 synonym-pairs of Stack Overflow and investigated the performance of ranking the suggested synonyms. Second, we surveyed 20 Android app developers to evaluate the synonym-suggestions of 10 randomly selected Android related tags.

Accuracy and performance of TSST

We first applied TSST to all the tags of the synonym-pairs and for each tag output a ranked list of suggestions. We then iterated the list of suggested synonyms for each tag to check whether a correct suggestion was found within the top 1, top 3, top 5, top 10, or top 15 suggestions. The reference set consists of source and target tags of each created synonym-pair and each synonym-pair is evaluated twice. Therefore, we divided the number of correct suggestions by 2. Figure 4 presents the numbers of correctly matched synonym-pairs within the top n suggestions and the number of correct found synonyms, disregarding the rank of the correct suggestion.

Overall, 2,455 out of 2,765 of the synonym-pairs were found by TSST, disregarding on which position the correct synonym was. This is an accuracy of 88.4%. Out of the 2,455, 1,839 (74.9%) were within the top 15 suggestions. For 1,766 (71.9%) tags the correct synonym tag was suggested within the top 10 suggestions. TSST found 1,660 (67.9%) synonyms within the top 5 suggestions, and 1,464 (59.8%) synonyms within the top 3 suggestions. Finally, 1,123 out of the 2,455 (45,9%) tag synonyms matched the first suggestion of TSST.

Online survey with Android app developers

To evaluate how TSST performs on a new set of posts, we applied it to 10 tags selected from Stack Overflow. We decided to focus on tags that are related to questions tagged with `android`. Furthermore, we selected the 10 tags based on the distribution of the number of posts tagged with Android related tags. The distribution is presented in Figure 5.

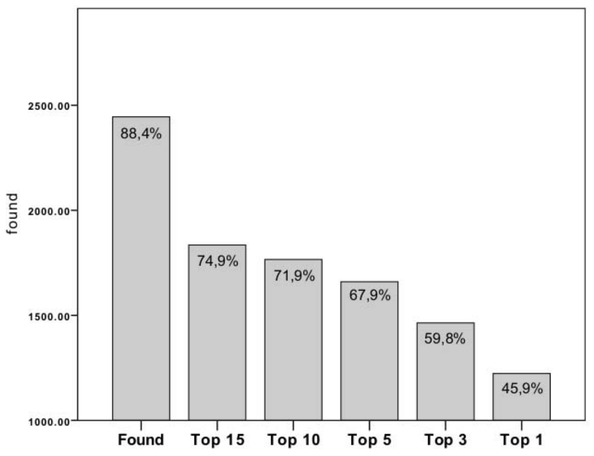

Fig. 4. Numbers of correctly suggested synonym-pairs from the Stack Overflow reference set within the top $n = \{1, 3, 5, 10, 15\}$ suggestions.

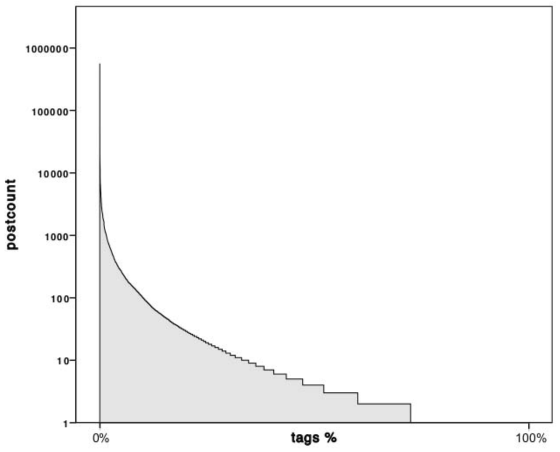

Fig. 5. Distribution of number of posts tagged with Android related tags (log-scale).

Since about 40% of the Android tags are used only once, we randomly selected 4 tags from this set of tags. 20% of the tags have a count between 3 and 6, therefore we randomly selected 2 tags from this set of tags. 10% of the tags are used 7 to 12 times, 10% have a count between 13 and 29, 10% are used between 30 and 100 times, and 10% of the tags have a count >=101. Consequently, we randomly selected one tag for each range to complete the set of 10 tags. The selected tags are: `automation`, `decompiler`, `interactive`, `case-insensitive`, `date-comparison`, `public-method`, `repository-pattern`, `redundant`, `spring-roo`, and `canonical-name`.

Table I shows the selected tags and their ranges. We used this set as input for TSST and set *maxSuggestions* to 15. The reason to select 10 tags and set *maxSuggestions* to 15 was to allow our study participants to fill in the survey in a reasonable amount of time. Each participant had a maximum $10 * 15 = 150$ synonym suggestions to rank that should not take more than 10 minutes. We wanted to make sure that participants were able to keep their attention throughout the survey reducing the risk for errors.

TABLE I
SELECTED ANDROID RELATED TAGS AND THEIR RANGES

#Tags	Tagname(s)	Range	%
1	`automation`	>100	10%
1	`decompiler`	30-100	10%
1	`interactive`	13-29	10%
1	`case-insensitive`	7-12	10%
2	`date-comparison, public method`	2-6	20%
4	`repository-pattern, redundant spring-roo, canonical name`	1	40%

The synonym suggestions output by TSST are evaluated in an online survey with 20 Android app developers from industry and university. The online survey consists of three parts. In the first part, we asked questions about the programming experience of the developers, such as years of experience in programming in Java, years of experience in programming Android apps and if they develop Android apps as part of their job. In the second part of the survey, we asked questions on whether and how they use Stack Overflow: do they use Stack Overflow for posting and/or answering questions or just reading posts. We also asked if they use tags, have already created a new tag, and know about the system of tag-synonyms on Stack Overflow. The third part of our survey consists of the 10 tags and for each tag a maximum number of 15 synonyms suggested by TSST. For each tag, the Android developers chose one synonym they see as an appropriate synonym or the option 'no suggestion fits'.

Regarding the answers to the first two parts of our survey, 75% of the developers have experience in programming in Java for more than 3 years. 40% developed Android apps for more than three years, 30% have developed Android apps for 1 to 3 years, and only 30% are professional Android app developers. The remaining 70% developed Android apps for private or university puroposes. 90% of the developers use Stack Overflow, about 25% of them ask and/or answer questions. None of the developers stated to have created a tag and only 15% know that there exist synonyms for tags.

Figure 6 represents the box-plots of the answers of the online survey. We provide detailed results of the questionnaire on our website.[9]

For the tag `decompiler` none of the developers chose 'no suggestion fits'. 85% of the developers agreed on the synonym `decompiling`. 95% of the developers found a fitting synonym for the tag `public-method` within the suggestions. 35% of them agreed on the synonym `method`, 30% selected the plural `methods`. 90% of the developers chose a synonym for `date-comparison`.

[9]http://serg.aau.at/bin/view/StefanieBeyer/TSST

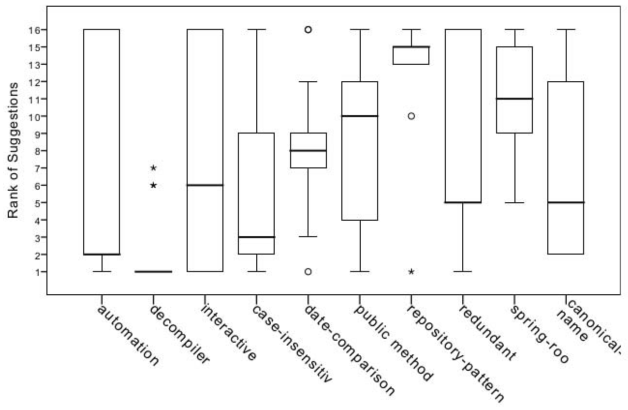

Fig. 6. Results of the Online Survey. 1 represents the first suggestion, 16 the option 'no suggestion fits'.

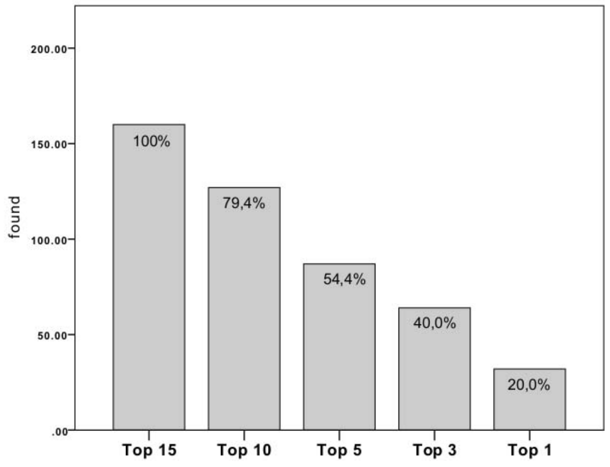

Fig. 7. Numbers of selected synonyms from the list of the top $n = \{1, 3, 5, 10, 15\}$ suggestions provided by TSST.

30% agreed on the suggestion `datetime-functions`, 25% on `datetime-operation`. For the tag `repository-pattern`, 85% of the developers found an adequate synonym. 40% agreed on the synonym `design-patterns`, 35% selected `repositories`. For the tag `spring-roo`, 45% of the developers agreed on the synonym `spring`. For the tag `canonical-name`, 30% agreed on the synonym `canonical`. For the tag `case-insensitive`, 75% of the developers found an adequate synonym within the list of suggested synonyms. 40% selected the synonym `case-sensitivity` and 20% the synonym `case-sensitive`. For the tags `automation` and `interactive`, 70% of the developers selected a synonym. For `automation`, 50% of the developers selected `automated-testing` and 10% selected the similar word `automated-tests`. For the tag `interactive`, 30% selected the first suggestion `f#-interactive`. However, also 25% of the developers selected 'no suggestion fits'. Only 60% of the developers found a fitting synonym for the tag `redundant` in the list of suggested synonyms. Out of these developers 45% agreed on the synonym `recurrence`.

Overall, we got 10 x 20 = 200 ratings with 160 (80%) selected synonyms. The option 'no suggestion fits' was selected only 40 times. 85% of the developers selected 'no suggestion fits' at least once. For 20% of the ratings the adequate synonym was not provided within the first 15 suggestions. In 32 out of 160 (20.0%) ratings the first synonym suggestion was selected. In 64 out of 160 (40.0%) ratings a synonym within the top 3 suggestions was selected. In 87 (54.4%) ratings the selected synonym was within the top 5 suggestions, and in 127 (79.4%) ratings it was within the top 10 suggestions.

Concerning the reliability of the ratings, we computed the intra-class correlation [10] over all ratings. There are 3 types of intra-class correlation and we chose the second one $ICC2k$, since in our survey each developer rated the suggestions for 10 randomly selected tags. The intra-class correlation on average for ratings on all tags is $ICC2k = 0.90$ that means strong reliability.

We also calculated the inter-rater agreement to evaluate the suggestions generated by TSST. These suggestions are ranked by TSST and so we have ordered-category data. Cohen's-Kappa [11] evaluates the inter-rater agreement of ordered-category data for two raters. To apply this statistics for more than two raters, we computed the weighted kappa for each pair of raters and calculated the average on all suggestions. We achieved $\kappa = 0.61$ that means substantial agreement.

To investigate how TSST performs on the randomly selected tags, we compare the rankings of correctly suggested synonyms of the online survey with our previous results achieved from applying TSST to the approved set of Stack Overflow synonym-pairs. For this we need to compare the results on the same base, namely top 15 suggestions. Therefore, we computed the ratios obtained in the first study for correctly suggested synonym-pairs relative to the top 15 suggestions. Relative to the top 15 suggestions, 96.6% out of the 1,839 correctly suggested synonyms were within the top 10 suggestions, 90.3% were within the top 5 suggestions, 79.6% within the top 3 suggestions and for 61.1% the first suggestion was the correct synonym. Comparing these numbers with the results of the online survey shown in Figure 7, we see that TSST performs significantly better on the approved set of Stack Overflow synonym-pairs.

Answering research question RQ3, applying TSST to the tags of the synonym-pairs of Stack Overflow, we achieved an accuracy of 74.9% for suggestions within the top 15 rankings and an accuracy of 45.9% for the first suggestion. Applying TSST to the 10 selected Android related tags, we achieved an accuracy of 80% for suggestions within the top 15. However, only 20% of the suggestions on the first rank were correct. Furthermore, the statistics of intra-class correlation and inter-rater agreement show that the reliability of the rating is strong and the raters achieved a substantial agreement on their ratings.

VI. Discussion

In this section, we first summarize the results of our experiments. Then, we discuss the differences in the accuracy obtained by TSST with the approved synonym-pairs of Stack Overflow and the online survey. Finally, we briefly discuss potential applications of our approach.

In this research, we investigated the synonym-pairs of Stack Overflow and derived 9 strategies to recreate the synonym-pairs systematically. We automated the strategies and developed the tag synonym suggestion tool TSST. At the time of writing, we have implemented 8 out of the 9 strategies. With these strategies, we were able to recreate 88.4% of the synonym-pairs. To estimate the accuracy of TSST, we applied the tool to the tags of the Stack Overflow synonym-pairs, as well as to 10 randomly selected Android related tags whose synonym suggestions were evaluated in an online survey with 20 Android app developers. The evaluation with the synonym-pairs showed an accuracy of 74.9% for suggesting a correct tag within the top 15 suggestions. The results of the online survey showed that Android developers found an adequate synonym within the top 15 suggestions in 80% of the ratings.

Comparing the results of the two evaluations, however, we found that TSST performs significantly better on the approved set of Stack Overflow synonym-pairs than on the new tags randomly selected among Android-related tags. In particular, when comparing the accuracy for suggesting the correct tag at the first position, the accuracy for the approved set is 45.9% compared to an accuracy of 20% for the 10 new tags. These differences indicate that the ranking of suggestions for new tags should be improved. Having a closer look at the suggestions and their rankings, we found that one reason for the lower accuracy concerns the similarity within the list of suggested synonyms. For instance, for the tag `public-method`, 7 developers selected the synonym `method`, 6 selected the similar synonym `methods`. While the synonym `method` was suggested on rank 12, the synonym `methods` was suggested on rank 4. Ideally, we argue, that these two suggestions should be ranked next to each other. In order to improve the ranking, we plan to consider these similarities in the ranking of suggestions, for instance, by apply stemming to the suggested synonyms.

Applying TSST to *all* tags of Stack Overflow, we expect to achieve good results for 15 suggestions per tag. Although not all strategies are implemented, yet, we estimate that the majority of the synonyms can be found. In the case of Android-related tags we expect to find the correct synonyms for 80% of the tags within the top 15 suggestions of TSST. This is based on the accuracy we obtained with the approved set of Stack Overflow synonym-pairs. Consequently, TSST could be used by privileged users to suggest and add synonym-pairs on Stack Overflow.

Furthermore, since there are too many tags with a similar meaning and because tags are often too detailed to be used in studies, we can use the knowledge about the synonyms of tags to group tags and reduce the amount of redundant tags. Having a less detailed and less redundant set of tags, we could improve studies investigating the categories of posts or trends and topics based on tags, such as [3] of Barua *et al.* and [4] of Treude *et al.*.

Threats to Validity

Threats to *internal validity* concern the manual analysis of the Stack Overflow synonym-pairs based on which we developed the 9 strategies for suggesting tag synonyms. To address this threat, we plan to evaluate the strategies with active users of Stack Overflow. Another threat to *internal validity* concerns errors due to losing attention when filling in the online survey. We addressed this threat by limiting the number of tags to 10 tags and the maximum suggestions to 15. We found that the time needed to fill the survey does not exceed 10 minutes.

Threats to *external validity* concern the focus of our study on Android-related tags, as well as the selection of 10 tags for our online survey. Regarding the first threat, we think that the amount of Android-related tags is sufficiently large and that our findings can be generalized to other mobile application platforms. Regarding the online survey, we randomly selected the set of tags to mitigate this threat, however, we are aware of that more tags from different domains need to be studied. Furthermore, the online survey needs to be performed with a larger group of users. We plan to address this in our future work.

VII. Related Work

In the last years, tags got increasing interest in research. There exist many approaches to suggest or recommend tags, for blogs, news sites, photo services, software artifacts or to find similar applications.

Zangerle *et al.* [12] focused on the recommendations of hashtags for Twitter, considering the text a user enters. Sigurbjörnsson *et al.* [13] developed `twofold` to recommend tags for online photo services, such as Flickr. Wang *et al.* [2] developed an auto tagging system for web pages, such as news sites, or blogs, as well as a tag suggestion system. These approaches are based on kNN and tdf-idf. Al-Kofahi *et al.* [14] focused on the recommendation of tags for software artifacts using fuzzy set theory.

Thung *et al.* [15] used collaborative tagging to detect similar applications on SourceForge. Wang *et al.* [16] investigated also similar tags and their taxonomy, and created a hierachy of tags. Tian *et al.* [17] developed $WordSim^{SE}$, a lexical database, to identify similar words in software engineering context.

Even more approaches deal with the recommendation of tags for software information sites, such as Stack Overflow. Saha *et al.* [18] also investigated an approach to suggesting tags for posts on Stack Overflow automatically. They use SVM (Support Vector Machine) and are able to predict missing tags. Wang *et al.* [19] developed `EnTagRec`, an approach to predict tags for software information sites, such as Stack Overflow or Ask Ubuntu. They use historical tag assignments to predict tags with labeled LDA. Stanley *et al.* [20] implemented a tag prediction system for the dataset of Stack Overflow. Their approach uses the Bayesian probabilistic model based on ACT-Rs declarative memory retrieval mechanisms. It may be used to introduce new tags to the author, as well as to find tags used wrongly. Xia *et al.* [21] introduced `TagRec`, a tag recommender system for Stack Overflow using tdf-idf, binary relevance, and naive Bayes. Furthermore, Short *et al.* [22] developed `NetTagCombine` to recommend tags for posts on Stack Overflow with tdf-idf. For their recommendation process they take into account the synonyms of tags provided by Stack Overflow.

These approaches mainly predict tags for posts based on quantitative methods and heuristics. In contrast, TSST suggests synonyms for tags, based on the strategies we derived from the manual investigation of the synonym-pairs of Stack Overflow.

Tags of Stack Overflow have also been used in previous studies to investigate approaches to predict if a post will be closed, for the categorization of posts, and topic finding. Galina *et al.* [23], as well as Correa *et al.* [24], used tags to predict closed questions on Stack Overflow. Treude *et al.* [4] used among other information the 200 most frequently used tags to investigate the topics discussed on Stack Overflow. Kavaler *et al.* [25] also used tags as additional information to find classes in the text of posts to link them to code. Parnin *et al.* [26] used tags to investigate the crowd documentation and API discussions on Stack Overflow. Barua *et al.* [3] explored the topics and trends on Stack Overflow over the time. For this, they also investigated the tags but found that the tags are too detailed and provide too much information for their reasons. These approaches motivate our approach to group tags by their synonyms, as suggested by TSST, thereby reducing the number of tags and amount of information.

Treude *et al.* [1] and Storey *et al.* [27] investigated the social and technical aspects of tagging, as well as the role tags, as part of social media, play in software engineering.

The investigation of related work shows there is a significant amount of research on recommending tags for blogs, news sites, as well as for Stack Overflow. Furthermore, tags have been used in studies as additional information for analyzing trends and topics on Stack Overflow. However, as pointed out by previous research, most prominent the work by Barua *et al.* [3], the tags of Stack Overflow are often too detailed and fine grained to consider them for a categorization. Although Stack Overflow provides a manual approach to reduce the amount of tags by creating synonyms, the proportion of synonyms to tags on Stack Overflow is low. Currently, there is no research on how the synonym-pairs are created and how this could be automated. TSST aims to fill this gap providing strategies to automate the creation of tag synonyms.

VIII. Conclusion

There exist more than 38,000 diverse tags on Stack Overflow but there are only 2,765 synonym-pairs that have been manually created by privileged Stack Overflow users. In order to increase the number of synonyms and reduce the amount of diverse tags on Stack Overflow, we investigated an approach to automate the suggestion of tag synonyms.

We first manually analyzed the synonym-pairs of Stack Overflow and found 9 strategies how these synonym-pairs are built. Then, to automate tag synonym suggestion, we implemented each strategy into TSST, a tag synonym suggestion tool that outputs a ranked list of synonym suggestions for a given input tag.

We first evaluated TSST with the approved set of Stack Overflow synonym-pairs. 2,445 out of 2,765 Stack Overflow synonym-pairs (88.4%) were found by TSST. Out of these, 1,660 (67.9%) were within the top 5 suggestions and for 1,123 out of 2,445 (45.9%) the first suggestion was the correct one. We further evaluated TSST with an online survey and asked 20 Android app developers for 10 tags to select one synonym out of the list of suggested synonyms. For 80% of the selected tags the developers found an adequate synonym within the top 15 suggestions. The inter-rater agreement statistics intra-class correlation and weighted Cohen's Kappa showed strong reliability of the ratings and substantial agreement among the raters.

Based on these results, we found that TSST could be used by privileged users to suggest and add synonym-pairs on Stack Overflow. We furthermore found that TSST can reduce the amount of redundant tags and thereby improve previous studies investigating the categories of posts or trends and topics based on tags.

Future work is concerned with improving TSST to more accurately suggest and rank of tag synonyms, as well as to check if there exist any valid synonyms. For instance, we plan to implement the strategy *Abbreviation/Synonym*, as well as extending the *Synonym-In-Word* strategy to find words in words. We will also consider the transitivity of tags in providing synonym-suggestions, as well as the relation of source tags and target tags. We plan to improve the ranking of the suggested synonyms by combining several strategies, such as *Metaphone* and *Similarity*. Furthermore, we plan to extend TSST to learn from existing, approved synonyms to suggest synonyms more accurately. An evaluation of TSST on a larger set of tags of Stack Overflow with more developers is also part of future work, as well as applying TSST to other Q&A sites of Stack Exchange.

References

[1] C. Treude and M.-A. Storey, "How tagging helps bridge the gap between social and technical aspects in software development," in *Proceedings of the International Conference on Software Engineering*. IEEE Computer Society, 2009, pp. 12–22.

[2] J. Wang and B. D. Davison, "Explorations in tag suggestion and query expansion," in *Proceedings of the workshop on Search in social media*. ACM, 2008, pp. 43–50.

[3] A. Barua, S. W. Thomas, and A. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, pp. 1–36, 2012.

[4] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?: Nier track," in *International Conference on Software Engineering*. IEEE, 2011, pp. 804–807.

[5] M. F. Porter, "An algorithm for suffix stripping," in *Readings in Information Retrieval*, K. Sparck Jones and P. Willett, Eds. Morgan Kaufmann Publishers Inc., 1997, pp. 313–316.

[6] P. Jaccard, "The distribution of the flora in the alpine zone," *New phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[7] V. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

[8] G. Kondrak, "N-gram similarity and distance," in *String Processing and Information Retrieval*, ser. Lecture Notes in Computer Science, M. Consens and G. Navarro, Eds. Springer, 2005, vol. 3772, pp. 115–126.

[9] L. Philips, "Hanging on the metaphone," *Computer Language*, vol. 7, no. 12, 1990.

[10] P. E. Shrout and J. L. Fleiss, "Intraclass correlations: uses in assessing rater reliability." *Psychological bulletin*, vol. 86, no. 2, p. 420, 1979.

[11] J. Cohen, "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit." *Psychological bulletin*, vol. 70, no. 4, p. 213, 1968.

[12] E. Zangerle, W. Gassler, and G. Specht, "Using tag recommendations to homogenize folksonomies in microblogging environments," in *Proceedings of the International Conference on Social Informatics*. Springer-Verlag, 2011, pp. 113–126.

[13] B. Sigurbjörnsson and R. van Zwol, "Flickr tag recommendation based on collective knowledge," in *Proceedings of the International Conference on World Wide Web*. ACM, 2008, pp. 327–336.

[14] J. Al-Kofahi, A. Tamrawi, T. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, "Fuzzy set approach for automatic tagging in evolving software," in *Proceedings of the International Conference on Software Maintenance*. IEEE, Sept 2010, pp. 1–10.

[15] F. Thung, D. Lo, and L. Jiang, "Detecting similar applications with collaborative tagging," in *International Conference on Software Maintenance*. IEEE, Sept 2012, pp. 600–603.

[16] S. Wang, D. Lo, and L. Jiang, "Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging," *2013 IEEE International Conference on Software Maintenance*, vol. 0, pp. 604–607, 2012.

[17] Y. Tian, D. Lo, and J. Lawall, "Automated construction of a software-specific word similarity database," in *Software Evolution Week, Conference on Software Maintenance, Reengineering and Reverse Engineering*. IEEE, Feb 2014, pp. 44–53.

[18] A. K. Saha, R. K. Saha, and K. A. Schneider, "A discriminative model approach for suggesting tags automatically for stack overflow questions," in *Proceedings of the International Workshop on Mining Software Repositories*. IEEE Press, 2013, pp. 73–76.

[19] S. Wang, D. Lo, B. Vasilescu, and A. Serebrenik, "Entagrec: An enhanced tag recommendation system for software information sites," in *International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 291–300.

[20] C. Stanley and M. D. Byrne, "Predicting tags for stackoverflow posts," in *Proceedings of the International Conference on Cognitive Modelling*, 2013, pp. 414–419.

[21] X. Xia, D. Lo, X. Wang, and B. Zhou, "Tag recommendation in software information sites," in *Proceedings of the Working Conference on Mining Software Repositories*. Piscataway, NJ, USA: IEEE Press, 2013, pp. 287–296.

[22] L. Short, C. Wong, and D. Zeng, "Tag recommendations in stackoverflow," 2014.

[23] E. G. Lezina and A. M. Kuznetsov, "Predict closed questions on stackoverflow," in *Proceedings of the Spring Researchers Colloquium on Database and Information Systems*, 2013, pp. 10–14.

[24] D. Correa and A. Sureka, "Fit or unfit: analysis and prediction of 'closed questions' on stack overflow," in *Proceedings of the Conference on Online social networks*. ACM, 2013, pp. 201–212.

[25] D. Kavaler, D. Posnett, C. Gibler, H. Chen, P. Devanbu, and V. Filkov, "Using and asking: Apis used in the android market and asked about in stackoverflow," in *Social Informatics*. Springer, 2013, pp. 405–418.

[26] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, "Crowd documentation: Exploring the coverage and the dynamics of api discussions on stack overflow," Citeseer, Tech. Rep., 2012.

[27] M.-A. Storey, C. Treude, A. van Deursen, and L.-T. Cheng, "The impact of social media on software engineering practices and tools," in *Proceedings of the Workshop on Future of Software Engineering Research*. ACM, 2010, pp. 359–364.