2016 IEEE/ACM 13th Working Conference on Mining Software Repositories

Grouping Android Tag Synonyms on Stack Overflow

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

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

ABSTRACT

On Stack Overflow, more than 38,000 diverse tags are used to classify posts. The Stack Overflow community provides tag synonyms to reduce the number of tags that have the same or similar meaning. In our previous research, we used those synonym pairs to derive a number of strategies to create tag synonyms automatically.

In this work, we continue this line of research and present an approach to group tag synonyms to meaningful topics. We represent our synonyms as directed, weighted graphs, and investigate several graph community detection algorithms to build meaningful groups of tags, also called tag communities.

We apply our approach to the tags obtained from Android-related Stack Overflow posts and evaluate the resulting tag communities quantitatively with various community metrics. In addition, we evaluate our approach qualitatively through a manual inspection and comparison of a random sample of tag communities. Our results show that we can cluster the Android tags to 2,481 meaningful tag communities. We also show how these tag communities can be used to derive trends of topics of Android-related questions on Stack Overflow.

CCS Concepts

•**Information systems** → *Clustering;*

Keywords

Stack Overflow, Tags, Android, Clustering

1. INTRODUCTION

Stack Overflow provides tagging of questions, a service of Web 2.0 to informally label and classify data [23, 25]. In the last years, researchers have performed various studies with the posts of Stack Overflow. For instance, they investigated trends and topics [3], the reasons for closed or unanswered questions [6, 10, 22], as well as question types and problem types discussed on Stack Overflow [20, 4]. Often they used tags as a starting point for their studies. However, as found by Barua *et al.* [3], the tags of Stack Overflow are too fine-grained and often have the same or similar meaning. Addressing this problem, Barua *et al.* [3] manually grouped related tags to meaningful topics. Similarly, Rosen *et al.* [20] also started with a manually selected group of related tags to study topics of Android app development. While manually grouping a small set of tags is feasible, we argue that grouping more than 38,000 tags, which is the number of tags on Stack Overflow at the time of September 2014, is not.

Regarding the similar meaning of tags, the community of Stack Overflow allows privileged users with a reputation of $> 1,500$ to suggest *tag synonyms* to reduce the amount of redundant tags. Stack Overflow defines: *"A tag synonym is usually a tag that has exactly the same meaning as some other tag, such as* `algorithm` *and* `algorithms`*. In some cases, tags that are* subsets *of other tags will also be considered synonyms, such as* `java-se` *for* `java`*."*[1] If other privileged users vote for a synonym, it is approved and may be used for tagging. However, in September 2014 there were only 2,765 manually suggested tag synonym pairs on Stack Overflow which is a small amount compared to more than 38,000 existing tags on Stack Overflow.

In our previous research [5], we analyzed the approved tag synonym pairs created by the Stack Overflow community and derived several strategies, implemented in the TSST approach, to compute synonym pairs automatically. In this paper, we continue this line of research and aim at grouping related tags, meaning tag synonym pairs, to topics automatically. The main research question addressed in this paper is: *To which extent can we group tag synonym pairs to meaningful topics using an automated approach?*

To automate the approach, we represent tag synonym pairs as a directed, weighted graph and use clustering and community detection algorithms to group them. Since there exist many clustering and community detection algorithms, we first investigate which algorithms fit our needs best. We apply the set of algorithms provided by the `igraph` package [7] of *R* to our data set. The data set consists of the tag synonym pairs provided by Stack Overflow and the tag synonym pairs obtained by applying TSST to Android-related tags on Stack Overflow. We find that the *walktrap* community detecting algorithm fits our needs best.

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

MSR'16, May 14-15, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2901739.2901750

Second, using the same data set, we perform several experiments with the walktrap community detection algorithm to determine the most appropriate configuration of the algorithm for grouping synonym pairs to tag communities. For this, we evaluate the resulting tag communities quantitatively and qualitatively. Regarding the quantitative evaluation, we measure several properties of tag communities, such as the maximum and minimum sizes of communities, and the number of communities of size one, two, or greater than 30. For the qualitative evaluation, we manually investigate the communities of randomly selected tags obtained by various configurations of the walktrap algorithm. Combining the results of both evaluations, we find that using tag synonym pairs with a ranking ≥ 0.55 as input and setting the step size of the walktrap algorithm to 3 outputs the most meaningful communities. Finally, using this configuration, we show an application of our approach to derive trends of hot topics related to Android app development.

Summarized, this paper makes the following contributions:

- A comparison of community detection algorithms to find meaningful communities of tag synonyms on Stack Overflow
- A quantitative and qualitative evaluation of the communities obtained by various configurations of the algorithm and input data
- An application of the results by deriving trends of hot topics of Android-related posts on Stack Overflow

The remainder of this paper is organized as follows. In Section 2, we briefly introduce TSST and discuss the structure of tag synonym pairs. In Section 3, we introduce the definitions of communities and tag communities and present the criteria for selecting a community detection algorithm. In Section 4 and Section 5, we evaluate the various configurations of the community detection algorithm and the resulting tag communities. In Section 6, we apply our approach to tag synonym pairs obtained from Android-related tags on Stack Overflow and in Section 7, we discuss the results and their limitations, as well as the threats to validity. Related work is presented in Section 8. In Section 9, we present conclusions and directions for future work.

2. TAG SYNONYM SUGGESTION TOOL

In this section, we briefly describe the Tag Synonym Suggestion Tool (TSST) for creating tag synonyms and the set of improvements made over the previous version of TSST presented in [5].

2.1 TSST Approach

In TSST, each tag synonym pair consists of a *target tag* and a *source tag*, representing the master-synonym relationship of tag synonyms on Stack Overflow. Note, the *source tag* has not necessarily the same meaning as the *target tag* and is also allowed to be a subset of the *target tag*. The *target tag* is defined to be the more generic tag and the *source tag* to be the more specific one.[1] A *target tag* may be the *source tag* of another synonym pair and *source tags* may also be *target tags* in other synonym pair relationships. Tag synonym pairs can build a hierarchy, however, synonym pairs are not bidirectional.

TSST generates tag synonym pairs through applying several strategies that we derived by manually analyzing the approved tag synonym pairs of Stack Overflow. For instance, TSST uses the strategy *Stemming* to match tags by their stem, such as `clustered-indexing` and `clustered-index`, or the strategy *Metaphone* to match tags by their pronunciation, such as `tchart` and `teechart`. Other strategies used by TSST are *Synonym-In-Tag, Synonym-In-Word, Similarity, Acronym, DotSharpMinusPlus, Abbreviations/Synonyms*, and *Numbers*. TSST applies the strategies in the order from restrictive to general. Starting with the most restrictive strategy *Stemming*, the strategies *Numbers, DotSharpMinusPlus, Synonym-In-Tag, Synonym- In-Word, Acronym, Similarity,* and *Metaphone* are applied. For more details on the strategies and TSST, we refer the reader to our paper [5].

With TSST we can generate tag synonym pairs of various quality, meaning that the *source tag* is actually a synonym or subset of the *target tag*. For this, we compute a ranking between 0.0 and 1.0 for each synonym pair. A ranking of 1.0 means the likelihood of two tags forming a valid synonym pair according to TSST is 100%. In the previous version of TSST, the ranking of the suggested synonyms is based on the count of strategies that output the same synonym pair. There we assumed, the more strategies generate the same synonym pair, the more likely it is indeed a valid tag synonym pair. In the evaluation with the 2,765 approved synonym pairs of Stack Overflow, TSST showed to correctly generate 88.4% of the synonym pairs.

2.2 TSST Improvements

Compared to the previous version of TSST presented in [5], we improved the ranking of tag synonym pairs, as well as several strategies to compute synonym pairs. Regarding the ranking, our manual evaluation showed that the quality of the tag synonym pairs mainly depends on the strategy used to create a synonym pair. For instance, a tag synonym pair created by the strategy *Stemming* is more likely to be a valid synonym pair than a synonym pair created with *Metaphone*, where the matching is based on the pronunciation of the words. Therefore, we refined the ranking of the suggested tag synonyms and based it mainly on the strategy used to build a tag synonym. Furthermore, we still consider the number of strategies that generate the same synonym pair and add the average position of the target tag in Stack Overflow posts to compute the ranking. Regarding the position of tags, we found that the first tag in a post is more general than the last one. For instance, the generic tag android is the first tag in more than 80% of the posts tagged with `android`.

Regarding the strategies, we refined the implementation of the strategies *Numbers* and *DotSharpMinusPlus* that in the new version cover more combinations and variations of tag synonyms. The strategies *Synonym-In-Tag, Synonym-In-Word,* and *Acronym* were also refined to improve the ranking that strongly depends on the strategies. We supplemented the strategy *Synonym-In-Tag* with *Synonym-in-Tag-Subtag* and *Synonym-In-Tag-Like*.

In the previous version, the tag synonym pairs `cluster-indexing` and `indexing`, `abstract-method` and `static-method`, and `html5-video` and `videoview` are all created by the strategy *Synonym-In-Tag*. In the new version, the first synonym pair is created with the original strategy *Synonym-*

In-Tag, the second one with *Synonym-In-Tag-Subtag*, and the last one with *Synonym-In-Tag-Like*.

We also supplemented the strategy *Synonym-In-Word* with *Synonym-In-Word-Like* to differ between the creation of tag synonym pairs, such as *qtmultimedia* and *time*, and *threading* and *multithreading*. A synonym pair is created with the original strategy *Synonym-In-Word*, if one tag starts or ends with an other tag, such as *threading* and *multithreading*. A synonym pair is created with *Synonym-In-Word-Like*, if a tag occurs in the middle of another word, such as *qtmultimedia* and *time*.

Finally, we supplemented the strategy *Acronym* with *Acronym-Start-Or-End* and *Acronym-Combination*. For instance, synonym pairs, such as *command-line-interface* and *cli*, and *user-interface* and *ui* belong to the original strategy *Acronym*. *webcontrol* and *webbrowser-control* are created with *Acronym-Combination*, since *webcontrol* is a combination of the first parts of the *pots* (part of tags) of *webbrowser-control*. The strategy *Acronym-Start-Or-End* matches tags where the acronym of one tag is the start or end of another tag, such as *ie7-bug* and *internet-explorer-7*.

The order in which the strategies are applied is the same as in the previous version of TSST. However, when calculating the ranking, the supplementing strategies get a lower ranking than synonyms generated by the original strategy, since we found that they often lead to synonym pairs of lower quality (see for instance the previous example *qtmultimedia* and *time*).

To evaluate the improvements of TSST, we randomly selected 100 tag synonym pairs computed with tags from Stack Overflow and spot-checked them manually. When calculating the coverage of the generated synonym pairs, the results showed that the new version generates 88.7% of the approved tag synonym pairs correctly. This is a small improvement of only 0.03%, however the supplementing strategies help to more accurately determine the ranking of tag synonym pairs.

3. TAG COMMUNITIES

Our goal is to group tag synonym pairs into meaningful topics. For this, our basic approach is to represent the tag synonym pairs obtained by TSST as a directed and weighted graph and apply clustering and community detection algorithms to the graph. In the following, we first introduce basic graph definitions, as well as definitions for communities and tag communities. Furthermore, we present the criteria for selecting community detection algorithms.

3.1 Definitions

We use the definitions of directed, weighted graphs and paths from Cuvelier *et al.* [8]. A *directed, weighted graph* $G = (V, E)$ consists of a non-empty set V of vertices, and a set E of directed edges between the vertices $E \subseteq (u, v)|$ $u, v \in V$. Each directed edge $(u, v) \in E$ has a start vertex u, and an end vertex v. We do not allow self-loops in our tag synonym pairs, so that $E \subseteq \{(u, v)|u, v \in V \wedge u \neq v\}$. Furthermore, a function $\omega : E \to \mathbb{R}$ exists that assigns a weight to each edge.

The *in-degree* of a vertex v, denoted as $deg^-(v)$, is the number of edges directed at v. A *subgraph* $G' = (V', E')$ of a graph $G = (V, E)$ is a graph such that $V' \subset V, E' \subset E$ and $(u, v) \in E'$ implies $u, v \in V'$. The graph G is a super-graph of G'. A subset C of V can define an induced subgraph $G(C) = (C, E(C))$, where $E(C) = \{(u, v) \in E|u, v \in C\}$. For a directed graph $G = (V, E)$, an integer $n \geq 0$, and vertices $u, v \in V$, a *path* (or walk) of length n from u to v in G is a sequence of vertices and edges $x_0, e_1, x_1, e_2, ..., x_n, e_n$, such that $x_0 = u$ and $x_n = v$, and such that $e_i = (x_{i-1}, x_i) \in E$ for all $i \in \{1, ..., n\}$. Two vertices v and u are called *neighbors* or adjacent, if they are connected by an edge. The set of neighbors of a node v, denoted $\Gamma(v)$, is called its *neighborhood*.

A *community* C is the result of an algorithm to split a graph G into subgraphs $G'(V', E')$ of vertices V' and edges E'. The vertices V' are strongly related to each other [9]. There exist various definitions for communities, since they are algorithmically defined. Fortunato found that the majority of these definitions agree on that *"there must be more edges "inside" the community than edges linking vertices of the community with the rest of the graph"* [9]. In addition, there also exists a *path* P between two vertices $u, v \in V$, of a community C that does not leave the community C [8].

Each tag synonym pair obtained by TSST consists of a *source-tag*, a *target-tag*, and a *ranking*. With this information, we can represent the tag synonym pairs as a directed and weighted graph $G(V, E, \omega)$. We define the union of tags of *source-tags* and *target-tags* as vertices V of a graph G. The direction of the edges E represent the synonym-master relationship from *source-tag* (synonym) to *target-tag* (master). The direction shows the hierarchy of the tag synonym pairs. The weighting function ω of graph G represents the *ranking* of tag synonym pairs.

Our goal is to split the graph G consisting of tag synonym pairs into groups of semantically related tags. In the following, we refer to these groups of tags as *tag communities*. A *tag community* is a community as defined by Fortunato, extended by adding that each community $C(V, E)$ consists of vertices V that are semantically related to each other. Each tag community consists of a generic community-name $comName$ and a number of tags. The number of tags denotes the *size* of a community. The $comName$, representing the topic of the tag community, is defined to be the name of the node v with the maximum degree $deg^-(v)$ of incoming edges.

3.2 Community Detection Algorithms

Analogue to the various definitions of communities, there are various algorithms for community detection in graphs. We search for a community detection algorithm that fulfills the following criteria:

- *data input size:* The algorithm is able to deal with graphs consisting of more than $10,000$ vertices.
- *determinism:* The algorithm is deterministic.
- *performance:* The algorithm terminates within one hour.
- *weighting:* The algorithm considers the weights of the edges of the input graph.
- *direction:* The algorithm considers the direction of the edges of the input graph.

To find an appropriate algorithm, we investigated the community detection algorithms provided by the `igraph` package for network analysis and visualization [7]. The straight forward approach to extract communities is to apply

Clustering. This algorithm computes independent subgraphs $G'(E', V')$ with no connection to other vertices V outside the subgraph. Applying this algorithm to the set of tag synonym pairs obtained from Android-related tags, we obtained one big cluster of size 14,613 and two clusters of size 5 and 2. Since we aim at having more fine-grained communities, and not one big cluster, we excluded this algorithm for computing tag communities.

The *walktrap community* algorithm of Pons and Latapy [19] is a hierarchical clustering algorithm. It takes directed and weighted graphs as input and calculates densely connected subgraphs (=communities) via random walks of a given step size. *"Two neighborhood nodes are more probable to belong on the same community, if they can be mutually visited by random walks starting from these nodes"* [13]. The default step size for random walks suggested by Pons and Latapy is 4. A detailed description of the algorithm can be found in [19]. The walktrap community algorithm deals with large graphs, is deterministic, and deals with weighted and directed graphs. Furthermore, applying the algorithm to our data, we find that also the performance criteria is satisfied. Therefore, this algorithm fulfills all of the predefined criteria and we selected it for our approach.

We investigated also other community detection algorithms provided by `igraph`, such as *edge-betweenness community*, *fastgreedy community*, *label-propagation community*, *leading-eigenvector community*, *multilevelcommunity*, *optimal community*, and *spinglass-community*. Since none of them fulfilled all our criteria we excluded these community detection algorithms. For instance, the *spinglass community* algorithm has a limit for the maximum number of communities, and the *optimal-community* algorithm is np-complete and only works on graphs with 500 or less vertices. The algorithms for *fastgreedy community*, *leading-eigenvector community*, *multilevel community*, and *label propagation* only work on undirected graphs. The latter even produces non deterministic results without an initial setting for each vertex. We also excluded the *edge betweenness community* algorithm for performance reasons. The calculation of communities on a graph with more than 10,000 vertices took more than five hours on a standard computer with and Intel Core i7 CPU and 16GB main memory.

4. EVALUATION OF WALKTRAP

In this section, we first present the data sets used for evaluating the walktrap community detection algorithm and then show the results obtained by running the algorithm with different configurations.

4.1 Data Sets

For the evaluation, we used the 12,717 tags obtained from Android-related posts on Stack Overflow as of September 2014. We then applied TSST to these tags and stored the 1,145,882 generated tag synonym pairs in a database. For each suggested synonym, we stored the *source-tag*, *target-tag*, and the *ranking*. Furthermore, the community of Stack Overflow provides a set of 2,765 approved tag synonym pairs. We added those synonym pairs to our data where either the *source tag* or *target tag* is contained in the set of Android-related tags, resulting in 2,261 additional synonym pairs and 1,907 additional tags. Since these tag synonym pairs have been approved by the Stack Overflow community, we set their ranking to 1.0. Furthermore, to avoid having a big community with the community-name `android`, we excluded all tag synonym pairs with the target tag `android`. Finally, we obtain a dataset consisting of 14,624 (12,717 + 1,907) distinct tags.

Table 1: Number of Tags and Tag Synonym Pairs of our Input Datasets

Input Datasets r_n	# Tags	# Tag Synonym Pairs
$r_{0.4}$	13,791	97,585
$r_{0.45}$	11,666	81,055
$r_{0.5}$	10,374	47,880
$r_{0.55}$	10,329	21,925
$r_{0.6}$	10,327	20,651
$r_{0.65}$	10,324	20,458

In the next step, we manually checked a random selected subset of tag synonym pairs regarding the quality of their ranking. We found that synonym pairs with a ranking $<$ 0.40 typically are not similar and do not show a synonym-master relationship. We filtered these synonym pairs resulting in a set of 97,585 tag synonym pairs formed by 13,791 different tags (94.30% of the 14,624 tags).

Since the ranking of tag synonym pairs might impact the computation of tag communities, we defined several input sets with different lower bounds of rankings to allow the analysis of this impact. In particular, we defined input sets with rankings ranging from $\geq$ 0.40 to $\geq$ 0.65 in step sizes of 0.05. We refer to the input sets as $r_n, n \in \{0.40, 0.45, 0.50, 0.55, 0.60, 0.65\}$, meaning the input set $r_{0.50}$ contains the tag synonym pairs having a ranking from 0.50 to 1.0. We selected $\geq$ 0.65 as an upper limit for the lower bound since these synonym pairs still cover a significant amount of the Android-related tags, namely 8,417 of 14,624 (57.56%). Table 1 shows the resulting number of tags and tag synonym pairs for each input set r_n. In the following, we refer to these input sets also as input graphs.

4.2 Configuration of the Walktrap Algorithm

The walktrap algorithm has two input parameters: The step size for the random walks and the input graph to split into communities. Additional to the default step size 4 for the random walk suggested by Pons and Latapy, we ran the community detection algorithm with step sizes of $2, 3, 5$, and 6. Regarding the input graph we used the data sets $r_n, n \in \{0.40, 0.45, 0.50, 0.55, 0.60, 0.65\}$ described in Section 4.1. To evaluate the impact of these two parameters on the resulting tag communities, we introduce a number of community *metrics*, ordered by priority p:

- $\#comSize1$: number of communities with size 1 (p_1)
- $\#comSize2$: number of communities with size 2 (p_2)
- $\#com > 30$: number of communities with size greater than 30 (p_3)
- $\#com$: number of communities (p_4)
- $maxSize$: maximum size of the communities (p_5)
- $medianSize$: median size of the communities (p_6)

Since our goal is to group tag synonyms to communities, our main criterion is to obtain communities consisting of more than one tag, thus the values for $\#comSize1$ should be

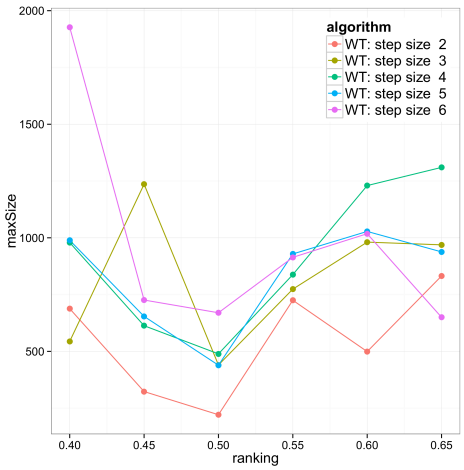


Figure 1: Maximum size of communities obtained by the walktrap community algorithm

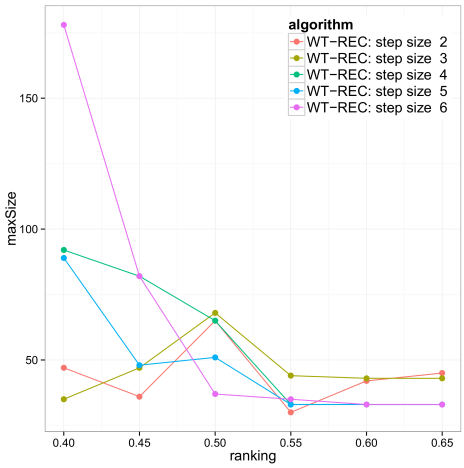


Figure 2: Maximum size of communities obtained by the recursive walktrap community algorithm (WT-REC)

low. Similarly, the number of communities with size 2 measured by $\#comSize2$, consisting of one tag synonym pair, should also be low. Furthermore, we aim at minimizing the number of communities $\#com$. The $maxSize$ of communities measured as number of tags contained by the largest community should be low. Ideally, the maximum size of communities should be 30 and the number of communities having size > 30 should be zero. We obtained the value 30 for this threshold by manually investigating a set of 300 communities computed with the six input sets and five step sizes for 10 randomly selected tags. The main author and a PhD student investigated separately 300 communities counting the number of tags semantically belonging to a community and the number of tags that do not. The results showed that communities with a size > 30 should be split into subcommunities. Finally, we aim at communities with many tags and therefore, the $medianSize$ should be maximized.

In a first experiment to investigate the maximum size of communities, we applied the walktrap community detection algorithm with step sizes from 2 to 6 to the input graphs $r_{0.40}$ to $r_{0.65}$. Figure 1 shows the maximum size of communities obtained for each step size and input graph. Depending on the ranking, the lowest maximum size of a community is 221 tags for step size 2 and input graph $r_{0.50}$ and the size of the largest community is $1,927$ tags for step size 6 and input graph $r_{0.40}$. These results clearly show that none of the used step sizes and input data sets leads to a result that satisfies our criteria for the maximum size of communities. To split the communities further into communities with a maximum size of 30, we came up with the approach to *recursively* apply the walktrap community detection algorithm until there was no remaining community with size > 30, or the community could not be split any more. We named this approach WT-REC.

Figure 2 shows the maximum size of communities computed with the WT-REC approach using the different step sizes and input graphs. Compared to the results shown in Figure 1 the maximum size of communities is significantly lower. In particular, for the input graphs from $r_{0.55}$ to $r_{0.65}$ the maximum size of communities is between 30 and 45. The charts in Figure 3 show several metrics of the communities obtained by WT-REC with step sizes 3, 4, and 5 (numerical values are listed in Table 2). They show that the values of the metrics $\#com$, $\#com > 30$, as well as $\#comSize1$ and $\#comSize2$ remain almost constant for the input graphs $r_{0.55}$ to $r_{0.65}$. Figure 4 shows that the step size mainly influences the values of $\#com$ and $\#comSize2$.

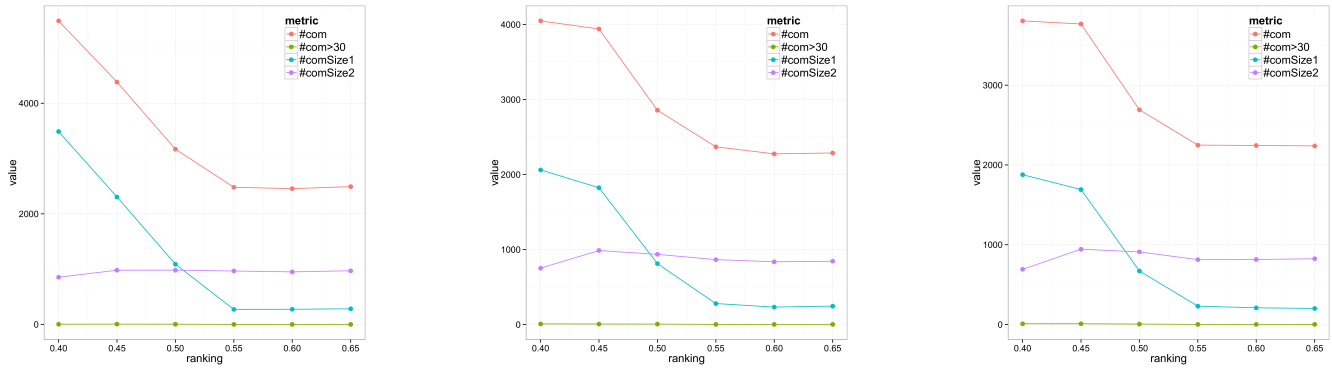
Based on these first results, we found that limiting the maximum size of communities to 30 tags and applying the walktrap community algorithm recursively, we can extract tag communities that satisfy our criteria. While the values of our community metrics differed for input graphs ranging from $r_{0.40}$ to $r_{0.55}$, we found that these differences were small between the input graphs $r_{0.55}$, $r_{0.60}$, and $r_{0.65}$. As a consequence, we only considered the tag communities computed from synonym pairs with ranking ≥ 0.55 as input graphs for our further evaluations.

5. EVALUATION OF THE WT-REC COMMUNITIES

In this section, we present the results of our quantitative and qualitative analysis of the tag communities computed with the WT-REC algorithm. The main goal of both evaluations was to determine the most appropriate configuration for WT-REC, in terms of step size and input graph, to obtain tag communities.

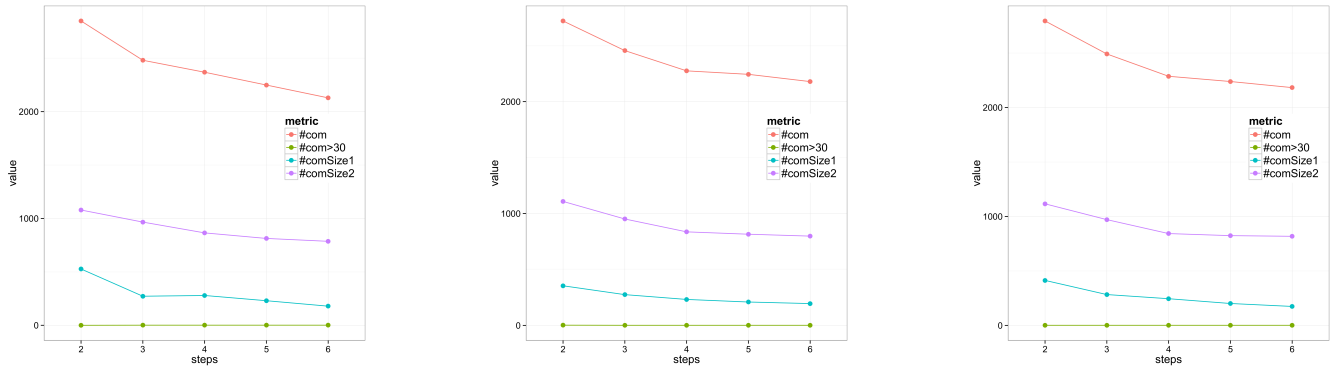
5.1 Quantitative Analysis of Communities

For the quantitative analysis, we evaluated the tag communities based on the metrics and their priorities as introduced in Section 4.2. Regarding the priorities we assigned a weight to each community metric whereas the weights sum up to 1.0. The most important metric is $\#comSize1$ with priority p_1. We assigned it the weight $w_1 = 0.4$, since we want to minimize the number of communities that only consist of a single tag. We assigned priority p_2 and the weight $w_2 = 0.3$ to the metric $\#comSize2$. We assigned priority p_3 and weight $w_3 = 0.2$ to the metric $\#com > 30$ and priority p_4 and weight $w_4 = 0.1$ to the metric $\#com$. We also investigated the metric $medianSize$ and found that the median size of communities is hardly influenced by the step



(a) Step size = 3 (b) Step size = 4 (c) Step size = 5

Figure 3: Tag communities computed with WT-REC for step sizes $3, 4$, and 5 for the input graphs $r_{0.40}$ to $r_{0.65}$



(a) Ranking ≥ 0.55 (b) Ranking ≥ 0.60 (c) Ranking ≥ 0.65

Figure 4: Tag communities computed with WT-REC for the input graphs $r_{0.50}$ to $r_{0.65}$ and step sizes 2 to 6

size and ranking therefore skipped this metric in our quantitative evaluation. Furthermore, we fixed the $maxSize$ of communities to 30 therefore also skipped this metric for our quantitative evaluation.

Using the weights and metric values presented in Table 2, we computed a score sc for each input graph r_k and step size s_t. The scores for the input graphs $r_k, k \in \{0.55, 0.60, 0.65\}$ are computed as follows:

$$sc_{r_k} = \sum_{j=1}^{|p|} |metric_{j,r_k}| * w_j$$

Regarding the score of an input graph r_k, we first count the number of step sizes in which that input graph obtained the best performance for a $metric_j$. Table 3 shows the counts for the different input graphs. For instance, the count for the metric $comeSize1$ is 1 for the input graph $r_{0.55}$ since only for step size 3 the metric value showed the best performance, meaning the lowest number of communities with only a single tag. The count for metric $comeSize1$ in the column $r_{0.60}$ is 2, since the two step sizes 2 and 4 showed the best performance for the input graph $r_{0.60}$. The score for an input graph r_k then is computed as the sum of the count computed for each metric denoted as $|metric_{j,r_k}|$ multiplied by the corresponding weight w_j of that metric.

Analogue, we calculate the score sc_{s_t} for each step size s_t. Table 4 shows the results of the scores for each step size. For instance, the count of $\#comSize1$ in the column 6 is 3, since the lowest value for $\#comSize1$ was achieved with step size 6 for all three input graphs $r_{0.55}$, $r_{0.60}$, and $r_{0.65}$. Similarly, the count for $\#comSize2$ and $\#com$ in column 6 is 3, since the best values for $\#comSize2$ and $\#com$ for all three input graphs were achieved using a step size of 6. Regarding the metric $\#com > 30$ the best performance was achieved with two input graphs per step size therefore the count for each step size is 2.

The results in Table 3 and Table 4 show that he highest scores are $sc_{r_k} = 2.5$ for the input graph $r_{0.60}$ and $sc_{s_t} = 2.8$ for the step size 6. These results suggest to use the input graph $r_{0.60}$ and step size 6 to run the walktrap community detection algorithm.

5.2 Qualitative Analysis of Communities

To evaluate the results qualitatively, we manually analyzed the communities generated for a subset of 35 Android-related tags from Stack Overflow. For each community, we counted the number of tags that semantically belong to the community and the number of tags that do not belong to the community. Based on these two measures, we define the *best* community as the community with the largest number

Table 2: Metrics of tag communities obtained with different step sizes and input graphs of WT-REC

step size	metric	input graph $r_{0.55}$	$r_{0.60}$	$r_{0.65}$
2	$\#comSize1$	527	354	413
	$\#comSize2$	1079	1108	1116
	$\#com > 30$	**0**	2	**1**
	$\#com$	2849	2721	2795
	$maxSize$	30	32	45
	$medianSize$	2	2	2
3	$\#comSize1$	272	275	283
	$\#comSize2$	966	951	971
	$\#com > 30$	1	**1**	**1**
	$\#com$	2481	2456	2492
	$maxSize$	44	43	43
	$medianSize$	3	3	2
4	$\#comSize1$	279	232	245
	$\#comSize2$	865	836	843
	$\#com > 30$	1	**1**	**1**
	$\#com$	2369	2275	2287
	$maxSize$	33	33	33
	$medianSize$	3	3	3
5	$\#comSize1$	230	209	201
	$\#comSize2$	813	815	824
	$\#com > 30$	1	**1**	**1**
	$\#com$	2249	2244	2239
	$maxSize$	33	33	33
	$medianSize$	3	3	3
6	$\#comSize1$	**179**	**195**	**174**
	$\#comSize2$	**786**	**798**	**818**
	$\#com > 30$	1	**1**	**1**
	$\#com$	**2129**	**2179**	**2183**
	$maxSize$	35	33	33
	$medianSize$	3	3	3

Table 3: Scores for each input graph

j	$metric_j$	w_j	input graph r_k 0.55	0.6	0.65
1	$\#comSize1$	0.4	1	2	2
2	$\#comSize2$	0.3	3	2	0
3	$\#com > 30$	0.2	5	4	4
4	$\#com$	0.1	1	3	1
		$sc_{r_k} =$	2.4	**2.5**	1.7

Table 4: Scores for each step size

j	$metric_j$	w_j	step size s_t 2	3	4	5	6
1	$\#comSize1$	0.4	0	0	0	0	3
2	$\#comSize2$	0.3	0	0	0	0	3
3	$\#com > 30$	0.2	2	2	2	2	2
4	$\#com$	0.1	0	0	0	0	3
		$sc_{s_t} =$	0.4	0.4	0.4	0.4	**2.8**

of tags that semantically belong to the community and the lowest number of tags that do not belong to the community. To reduce the bias of the manual evaluation, the first author and a PhD student separately evaluated the best communities for each tag. Both evaluators are familiar with Stack Overflow and are experienced with Android-related tags. We calculated Cohen's Kappa to measure the inter-rater agreement.

Table 5: Number of best communities for 21 tags using different combinations of input graph and step size

step size	input graph $r_{0.55}$	$r_{0.60}$	$r_{0.65}$	$\sum$
2	8	4	5	17
3	**11**	4	4	**19**
4	5	6	5	16
5	3	4	5	12
6	3	3	2	8
$\sum$	**25**	21	21	

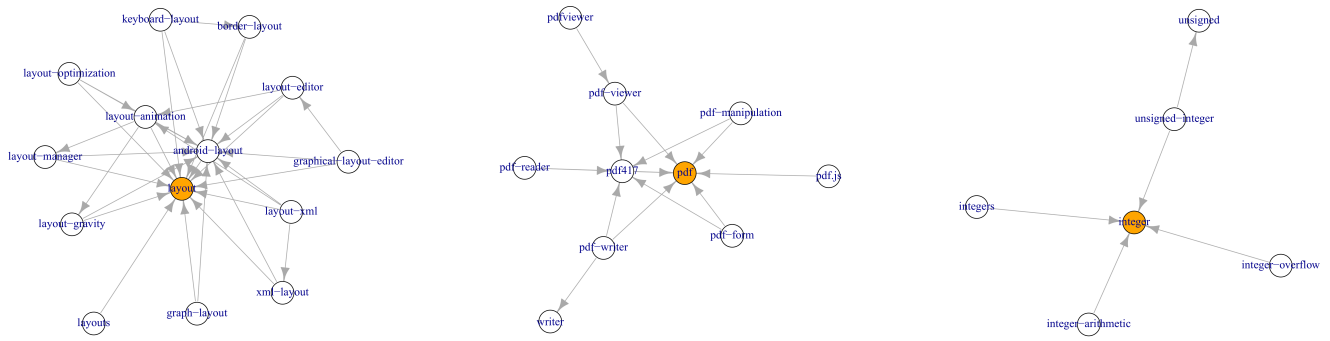
We started with the investigation of the 10 most frequently used tags for Android-related posts: `android-layout`, `eclipse`, `listview`, `android-intent`, `sqlite`, `android-activity`, `android-fragments`, `xml`, `json`, and `android-listview`. Furthermore, we randomly selected the following 25 Android-related tags: `ruby-on-rails`, `scheduling`, `interactive`, `django`, `paragraph`, `intervals`, `integer-arithmetic`, `seo`, `shpere`, `abstract`, `knox`, `transition`, `fortify`, `normalization`, `pdf`, `mpeg`, `row`, `cllocation`, `endpoint`, `undo`, `inflate-exception`, `type`, `buffered`, `Android-2.2`, and `jprogressbar`. For each tag, we investigated 15 communities for all combinations of rankings from ≥ 0.55 to ≥ 0.65 and step sizes 2 to 6 (in total $15 * 35 = 525$ communities).

Analyzing the communities, we found that the tag communities output for the 7 tags `paragraph`, `intervals`, `seo`, `sphere`, `knox`, `cllocation`, as well as `undo` in all settings consist of two nodes and one connecting edge. The number of communities of size 2 was not surprising since they account for up to 39.93% of the communities. Furthermore, the tag communities output for the 7 tags `sqlite`, `ruby-on-rails`, `scheduling`, `fortify`, `row`, `Android-2.2`, and `inflate-exception` consist of the same nodes and edges, independent of ranking and step size. We neglected these $14 * 15 = 210$ communities for our qualitative analysis, since they did not influence our results.

For each of the remaining 21 tags, each evaluator investigated the 315 communities and counted the number of best communities created per step size and input graph. Note, often more than one combination of step size and ranking results in the same best community. We compared the results from both evaluators using Cohen's Kappa and obtained a $\kappa = 0.88$, meaning almost perfect agreement between the two evaluators, according to the measurement of Landis *et al.* [14]. The results of this analysis are listed in Table 5.

According to the results in Table 5 the combination of step size 3 and input graph $r_{0.55}$ obtained the best tag communities for 11 tags. In contrast, using the input graphs $r_{0.60}$ and $r_{0.65}$ as well as a step size between 4 and 6 the best communities were obtained for only 2 to 6 tags. Also the count of the best results per step size (19 times) or ranking (26 times) showed that using step size 3 or the ranking $r_{0.55}$ lead in all combinations to the best communities.

Figure 5 shows examples of the communities for the tags `layout`, `pdf`, and `integer` to demonstrate the similarity of tags contained by a community computed with our approach. The yellow node represents the community name denoting the tag with the maximum degree of incoming edges.



(a) layout (b) pdf (c) integer

Figure 5: Tag communities of `layout`, `pdf`, and `integer` computed with WT-REC using step size 3 and input graph $r_{0.55}$

The tag community of `layout` shown in Figure 5a, consists of 14 tags and 32 edges between them, all of them belong semantically to the tag `layout`. This is supported by the fact that all tags contain the name `layout` as part of their name. The tag community `pdf` shown in Figure 5b consists of 10 tags and 14 edges. All tags, except of `writer` are semantically related to the tag `pdf`. Finally, Figure 5c shows the tag community `integer`, consisting of 6 tags and 5 edges between the tags. As for the other two examples, the names of the tags clearly suggest a semantic relationship with the tag community `integer`.

The results of the qualitative analysis differ from the results obtained by the quantitative analysis. While the quantitative analysis showed to use the input graph $r_{0.60}$ and step size 6 to compute tag communities, the qualitative analysis showed to use input graph $r_{0.55}$ and step size 3. The qualitative analysis also showed that the step size has more influence than the input graph, meaning the ranking. The more data we have for input, the more important is the step size to group relevant tags.

Since we aim at grouping tags to meaningful topics and we prioritize the accuracy of the topics over covering all tags, we favor the results of the qualitative analysis and consider using the input graph $r_{0.55}$ and step size 3 as the best setting for computing tag communities with WT-REC. Based on these results, we can answer our research question: *Using the recursive walktrap community detection algorithm on the input graph $r_{0.55}$ and setting step size to 3, we can group the 21,925 tag synonym pairs comprising 10,329 different tags into 2,481 tag communities with a median size of 3 tags. There is only 1 tag community consisting of 44 tags, 966 communities consist of two tags, and 272 of one tag.*

This result also means that our approach can significantly reduce the number of 10,329 different Android-related tags on Stack Overflow down to 2,481 tag communities (24.02%). The replication package of our study and the details of the analysis of the communities are provided on our website.[2]

6. APPLICATION OF RESULTS

In the last years, several researchers used tags to invesigate the topics and trends of questions on Stack Overflow. For example, Barua *et al.* [3] and Rosen *et al.* [20] manually grouped a manageable amount of semantically related tags to infer trends and topics on Stack Overflow. In this example, we show the advantages of our approach to perform this grouping of tags automatically.

Using the Stack Overflow dump from September 2014, we selected all 575,419 posts of Stack Overflow tagged with `android` and computed the tag synonym pairs with TSST. We then computed the input graph $r_{0.55}$ and used the WT-REC algorithm with step size 3 to compute the tag communities. Next, for each tag community, we replaced the contained tags with the community-name tag. We then counted for each tag community the number of new posts per month tagged with the community-name and stored the results in a database. In addition, we also counted the number of new posts per month for single tags.

The 10 most frequently used tag communities are `layout`, `listview`, `eclipse`, `fragment`, `sqlite`, `android-intent`, `android-activity`, `xml`, `service`, and `google-maps`. Figure 6 presents the trends of new posts per month for the 6 most frequently used tag communities. For each of the topics, the number of new posts per month (*postCount*) increased over time. There were only a few new posts per month for the topic `fragment` in the beginning of 2011. However, the interest in this topic increased and by the time of January 2014 it has been one of the most popular topics for new questions on Stack Overflow. The number of new posts per month peaked in the middle of 2014. The topic *layout* has been popular since the first posts on Stack Overflow about Android app development, which is also shown by the number of new posts per community in Table 6. The topic *layout* is followed by *listview* and *eclipse*. However, these topics were not as popular as *layout* in the beginning but increased since the middle of 2013.

To show the impact of using the tag communities, we compare our results to those achieved when only using single tags. Counting how frequently a tag is used in posts (*postCount*), we found that the tags `android-layout`, `eclipse`, `listview`, `android-intent`, `android-activity`, `sqlite`, `android-fragments`, `xml`, `android-listview`, and `json` are most frequently used. Comparing the post count of the top 10 most frequently used tags with the post count of the top 10 most frequently used tag communities, we found that

[2]http://serg.aau.at/bin/view/StefanieBeyer/Data_and_Tools

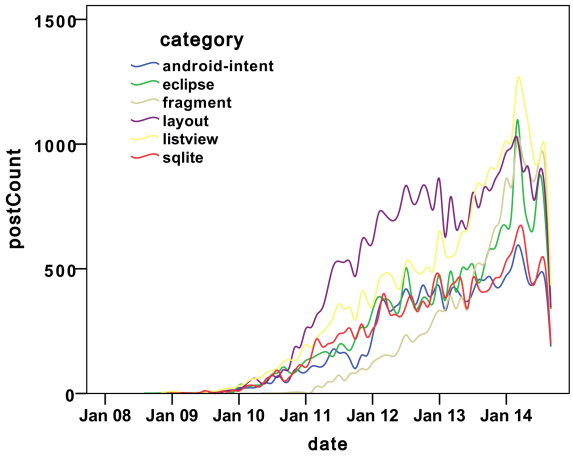


Figure 6: New posts per month per tag community

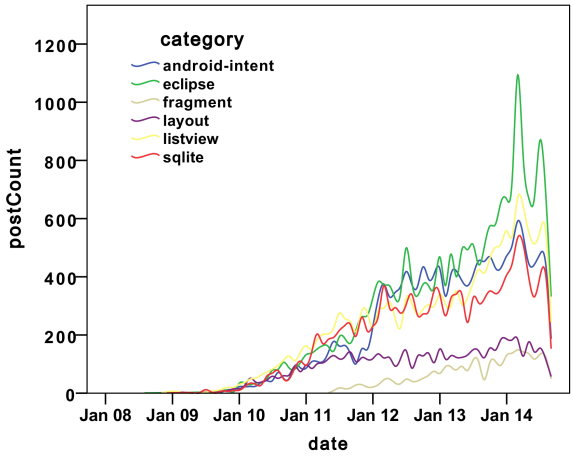


Figure 7: New posts per month per single tag

Table 6: Count of posts per tag community and tag

community	*count*	*tag*	*count*
layout	32,145	android-layout	25,487
		layout	6,474
listview	28,047	listview	16,590
		android-listview	11423
eclipse	20,161	eclipse	19,976
fragment	18,241	android-fragments	13,299
sqlite	17,144	sqlite	14,023
android-intent	15,697	android-intent	15,682
activiti	15,241	android-activity	14,251
xml	13,074	xml	11,908
service	13,022	service	4,928
google-maps	11,517	google-maps	7764

they have six topics in common. Using the tag communities, the tags `android-listview` and `listview` are grouped into the community `listview`, and `android-layout` and `layout` are grouped into `layout`. Table 6 shows that we obtained a significantly higher *postCount* using the tag communities for all topics.

Figure 7 shows the trends of new posts per month for the most frequently used tags. Comparing the trends in Figure 6 and Figure 7, we see that the most frequently used tag community `layout` was not as frequently used for tagging new posts when only considering single tags. Furthermore, the trend of the number of new posts per month for the community `fragment` shows a steeper increase because it groups tags, such as `android-fragments`, `fragment-lifecycle`, `headless-fragments`, and `page-fragments` into one topic. Similar observations can be made for the tag communities `listview`, `android-intent`, and `sqlite`.

Overall, the comparison clearly shows, when using our community approach, we obtain a more comprehensive and accurate picture of the trends of Android-related topics on Stack Overflow.

7. DISCUSSION

In this section, we summarize the results of our evaluations of tag communities and discuss the threats to validity.

7.1 Summary of Results

We investigated several community detection algorithms on graphs to group tag synonym pairs obtained by TSST to meaningful topics. Our results show that the walktrap community algorithm fits best. We applied the algorithm recursively to the graphs of tag synonym pairs with input sets $r_n, n \in \{0.40, 0.45, 0.50, 0.55, 0.60, 0.65\}$, covering 94.30% to 57.56% of all Android-related tags to obtain reasonable large communities. Furthermore, we varied the step size of the walktrap community algorithm from 2 to 6.

To quantitatively evaluate the tag communities, we calculated several community metrics, such as the maximum size of communities, or the number of communities with size 1 or 2. According to these metrics, we obtained the best results by using the input graph $r_{0.60}$ and and setting step size to 6. For the qualitative evaluation of the tag communities, we manually investigated 525 tag communities. In contrast to the quantitative analysis, the best results were achieved by setting the step size to 3 and using the input graph $r_{0.55}$. Summarizing the results of both analyses, we suggest to use the input graph $r_{0.55}$ and the step size 3 to compute tag communities with the walktrap community algorithm.

We presented one example of an application of our results to derive trends of Android-related tag communities on Stack Overflow. This shows that manual approaches to group tags and analyze topics, such as presented by Barua *et al.* [3] and Rosen *et al.* [20], can be automated with our approach. Furthermore, while in this paper we focus on analyzing Android-related tags, our approach can be easily applied to all tags on Stack Overflow.

7.2 Threats to Validity

Threats to *internal validity* concern the limitation of our selection of community detection algorithms provided by the `igraph` package of *R*. The algorithms we used assign each tag exactly to one community. To address this threat, we plan to investigate other community detection algorithms that support multiple assignments of tags to communities. Another threat to *internal validity* concerns the selection of the tag communities of the top 10 most frequently used Android-related tag communities for the manual inspection that may not be representative. To address this threat, we additionaly investigated the tag communities of 25 randomly selected Android-related tags. Another threat to *internal*

validity concerns the maximum size for communities set to 30. To mitigate this threat, we investigated 300 communities of various sizes and found that communities consisting of more than 30 tags should be split (see Section 4.2). Furthermore, the initial settings of the rankings for the input graphs could bias the results since not all of the Android-related tags are covered. However, we showed that the results are mainly influenced by the step size.

Threats to *external validity* concern the generalizability of the results obtained with the community detection algorithm on tag synonym pairs. We applied the walktrap community algorithm only to tag synonym pairs of Android-related tags and the tags of the approved synonym pairs on Stack Overflow. First, we think that the amount of Android-related tags is sufficiently large and covers diverse aspects of software engineering. Second, our approach can be easily applied to other data sources, such as the tags from Stack Exchange, to extend our studies.

8. RELATED WORK

In the past years, the interest in gaining information about posts, topics, and trends on Stack Overflow increased. For their investigations, researchers often used tags.

Correa *et al.* [6] and Lezina *et al.* [10] used tags to predict closed questions on Stack Overflow. Tags are also used as additional information by Kavaler *et al.* [12] to find classes in the posts' body and to link the posts of Stack Overflow to code. For the investigation of crowd documentation and API discussions, Parnin *et al.* [18] used the tags on Stack Overflow. Morrison *et al.* [17] investigated the tags of each user on Stack Overflow to find out if there is a relation between programming knowledge and age. Linares-Vasquez *et al.* [16, 15] used tags to filter posts related to mobile technologies, especially for Android. In our previous research, we also used tags to filter posts on Stack Overflow, that are related to Android app development [4]. To investigate the topics discussed on Stack Overflow and why questions do not get answered, Treude *et al.* [22] manually analyzed the 200 most frequently used tags.

Barua *et al.* [3] investigated the trends of topics on Stack Overflow over time. For this, they grouped related tags manually, such as `iphone, iphone-sdk-3.2, iphoneapp,` and `iphone-3gs`. They found that single tags are too detailed and provide too much information to us them for deriving trends and topics on Stack Overflow. Rosen *et al.* [20] approached to collect all tags related to mobile technology posts. They first grouped tags manually and then added all tags of posts that are tagged with a tag of the initial set of keywords. In contrast, we presented an approach to perform this grouping automatically on a large set of tags.

Tags have also been used in other research contexts. Wang *et al.* [26] presented a similarity metric for tags of software project hosting websites, such as Freecode. They characterized each tag by the document and its contents and defined the similarity between the tags by calculating the similarity between the documents. Furthermore, they applied the k-menoids clustering algorithm in multiple iterations to build a hierarchy of tag groups. Cui *et al.* [27] and Tsui *et al.* [24] investigated taxonomies and concepts of tags. Furthermore, Joorabchi *et al.* [11], as well as Bagheri *et al.* [2] mapped tags of Stack Overflow to Wikipedia concepts. In contrast to our approach, they use techniques, such as Latent Semantic Analysis on documents or machine learning, investigate the co-occurrences of tags, and have other objectives.

The investigation of related work shows that tags are fine-grained and inconsistent. Therefore, researchers started to group semantically related tags manually when using tags for their investigations. In our previous research [5], we presented TSST, a tag synonym suggestion tool that creates tag synonym pairs automatically. In this research, we continued this line of research and investigated community detection algorithms to group tag synonym pairs to tag communities that represent groups of semantically related tags. Although there are several approaches that use community detection algorithms on social network graphs, such as presented in [1] and [21], we are not aware of an approach applying community detection algorithms to tag synonym pairs to find meaningful topics. Using our approach, we can support researchers to group tags to semantically related topics to more accurately investigate, for instance the trends and topics on Stack Overflow with tags.

9. CONCLUSION AND FUTURE WORK

The community of Stack Overflow provides tag synonyms to reduce the number of 38,000 diverse tags on Stack Overflow. In our previous research [5], we investigated the tag synonym pairs of Stack Overflow and derived a set of strategies to create tag synonyms automatically.

In this paper, we aimed at grouping these synonym pairs to meaningful topics. For this, we represented the synonym pairs as directed, weighted graph and investigated various clustering and community detection algorithms to group tag synonyms. We experimented with various input graphs and settings for the algorithms and found that the *walktrap* community detection algorithm when applied recursively works best. Furthermore, we evaluated the tag communities quantitatively and qualitatively and found that the most meaningful tag communities can be obtained when using synonym pairs with a ranking ≥ 0.55 as input graph and setting the step size of the walktrap community algorithm to 3.

Using the tag communities, we showed an example in which we derived trends of hot topics related to Android-app development on Stack Overflow. The example clearly shows that the trends of new posts obtained with our tag communities differ from the trends obtained by only considering single tags, since our communities cover semantically similar tags. This has a significant impact on research and applications that rely on single tags and it is particularly this research and related approaches that can benefit the most from our tag communities.

Future work is concerned with the investigation of more community detection algorithms on graphs. In particular, we plan to investigate community detection algorithms supporting tags belonging to more than one community. We also plan to evaluate the algorithms on a larger set of tags of Stack Overflow. Furthermore, we plan to extend the investigation of trends of Android-related topics on Stack Overflow to further underline the value of tag communities. For instance, we will calculate several metrics obtained by posts that are grouped by tag communities.

10. REFERENCES

[1] A. Arenas, L. Danon, A. Diaz-Guilera, P. M. Gleiser, and R. Guimera. Community analysis in social networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):373–380, 2004.

[2] E. Bagheri and F. Ensan. Semantic tagging and linking of software engineering social content. *Automated Software Engineering*, 23(2):147–190, 2014.

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

[4] S. Beyer and M. Pinzger. A manual categorization of android app development issues on stack overflow. In *Proceedings of the International Conference on Software Maintenance and Evolution*, pages 531–535. IEEE, 2014.

[5] S. Beyer and M. Pinzger. Synonym suggestion for tags on stack overflow. In *Proceedings of the International Conference on Program Comprehension*, pages 94–103. IEEE, 2015.

[6] 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*, pages 201–212. ACM, 2013.

[7] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[8] E. Cuvelier and M.-A. Aufaure. *Business Intelligence: First European Summer School*, chapter Graph Mining and Communities Detection, pages 117–138. Springer, 2012.

[9] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[10] E. L. Galina and A. M. Kuznetsov. Predict closed questions on stackoverflow. *Proceedings of the Researchers Colloquium on Databases and Information Systems*, pages 10–14, 2013.

[11] A. Joorabchi, M. English, and A. E. Mahdi. Automatic mapping of user tags to wikipedia concepts: The case of a q&a website–stackoverflow. *Journal of Information Science*, pages 570–583, 2015.

[12] 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*, pages 405–418. Springer, 2013.

[13] D. Lai, H. Lu, and C. Nardini. Extracting weights from edge directions to find communities in directed networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(06):P06003, 2010.

[14] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[15] M. Linares-Vásquez, G. Bavota, M. Di Penta, R. Oliveto, and D. Poshyvanyk. How do api changes trigger stack overflow discussions? a study on the android sdk. In *Proceedings of the International Conference on Program Comprehension*, pages 83–94. ACM, 2014.

[16] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk. An exploratory analysis of mobile development issues using stack overflow. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 93–96. IEEE, 2013.

[17] P. Morrison and E. Murphy-Hill. Is programming knowledge related to age? an exploration of stack overflow. In *Proceedings of the Working Conference on Mining Software Repositories*, pages 69–72. IEEE, 2013.

[18] C. Parnin, C. Treude, and L. Grammel. Crowd documentation: Exploring the coverage and the dynamics of api discussions on stack overflow,. Technical report, Georgia Institute of Technology, 2012.

[19] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences*, volume 3373 of *Lecture Notes in Computer Science*, pages 284–293. Springer, 2005.

[20] C. Rosen and E. Shihab. What are mobile developers asking about? a large scale study using stack overflow. *Empirical Software Engineering*, pages 1–32, 2015.

[21] J. Scott. *Social network analysis*. SAGE Publications Ltd., 2012.

[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 International Conference on Software Engineering*, pages 804–807. IEEE, 2011.

[23] 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*, pages 12–22. IEEE, 2009.

[24] E. Tsui, W. M. Wang, C. F. Cheung, and A. S. Lau. A concept–relationship acquisition and inference approach for hierarchical taxonomy construction from tags. *Information processing & management*, 46(1):44–57, Elsevier, 2010.

[25] J. Wang and B. D. Davison. Explorations in tag suggestion and query expansion. In *Proceedings of the Workshop on Search in Social Media*, pages 43–50. ACM, 2008.

[26] S. Wang, D. Lo, and L. Jiang. Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging. In *Proceedings of the International Conference on Software Maintenance*, pages 604–607. IEEE, 2012.

[27] J. Yao, B. Cui, G. Cong, and Y. Huang. Evolutionary taxonomy construction from dynamic tag space. *World Wide Web*, 15(5-6):581–602, 2012.