Open access • Journal Article • DOI:10.1007/S00766-017-0274-X

An exploratory study of Twitter messages about software applications

Emitza Guzman, Rana Alkadhi, Norbert Seyff

Institutions: University of Zurich, Technische Universität München

Published on: 01 Sep 2017 - Requirements Engineering (Springer London)

Topics: Software evolution, Microblogging, Requirements engineering and Software

An Exploratory Study of Twitter Messages about Software Applications

Guzman, Emitza ; Alkadhi, Rana ; Seyff, Norbert

Abstract: Users of the Twitter microblogging platform share a considerable amount of information through short messages on a daily basis. Some of these so-called tweets discuss issues related to software and could include information that is relevant to the companies developing these applications. Such tweets have the potential to help requirements engineers better understand user needs and therefore provide important information for software evolution. However, little is known about the nature of tweets discussing software-related issues. In this paper, we report on the usage characteristics, content and automatic classification potential of tweets about software applications. Our results are based on an exploratory study in which we used descriptive statistics, content analysis, machine learning and lexical sentiment analysis to explore a dataset of 10,986,495 tweets about 30 different software applications. Our results show that searching for relevant information on software applications within the vast stream of tweets can be compared to looking for a needle in a haystack. However, this relevant information can provide valuable input for software companies and support the continuous evolution of the applications discussed in these tweets. Furthermore, our results show that it is possible to use machine learning and lexical sentiment analysis techniques to automatically extract information about the tweets regarding their relevance, authors and sentiment polarity.

DOI: https://doi.org/10.1007/s00766-017-0274-x

An exploratory study of Twitter messages about software applications

Emitza Guzman[1] · **Rana Alkadhi**[2] · **Norbert Seyff**[1,3]

Abstract Users of the Twitter microblogging platform share a considerable amount of information through short messages on a daily basis. Some of these so-called tweets discuss issues related to software and could include information that is relevant to the companies developing these applications. Such tweets have the potential to help requirements engineers better understand user needs and therefore provide important information for software evolution. However, little is known about the nature of tweets discussing software-related issues. In this paper, we report on the usage characteristics, content and automatic classification potential of tweets about software applications. Our results are based on an exploratory study in which we used descriptive statistics, content analysis, machine learning and lexical sentiment analysis to explore a dataset of 10,986,495 tweets about 30 different software applications. Our results show that searching for relevant information on software applications within the vast stream of tweets can be compared to looking for a needle in a haystack. However, this relevant information can provide valuable input for software companies and support the continuous evolution of the applications discussed in these tweets. Furthermore, our results show that it is possible to use machine learning and lexical sentiment analysis techniques to automatically extract information about the tweets regarding their relevance, authors and sentiment polarity.

Keywords Requirements engineering · Software evolution · User feedback · Content analysis · Textmining

1 Introduction

Twitter users write over five hundred million messages every day. Users discuss topics such as music, television, sports, politics and technology through these so-called *tweets*. This wide range of topics also includes software applications. Tweets about software applications could be similar to app reviews and discuss software failures and requests for new features [18, 26, 49]. Thus, tweets might be a relevant source of information for software companies. Stakeholders such as requirements engineers may benefit from these tweets, as they might allow them to better understand their users' needs and identify requirements relevant to the evolution of their software applications. Furthermore, tweets could allow for the collection of information from remote users, who are typically difficult to involve. The obtained insights may then be used to make informed decisions within software evolution processes.

However, little is known about the number and relevance of tweets regarding software applications and their impact on software evolution. In our research [22], we are the first to investigate the characteristics of tweets on software applications and their relevance to different stakeholders that are not necessarily developers (e.g., requirements engineers, project managers and users).

This manuscript is based on work published at the International IEEE Requirements Engineering Conference

✉ Emitza Guzman
guzman@ifi.uzh.ch

Rana Alkadhi
alkadhi@in.tum.de

Norbert Seyff
norbert.seyff@fhnw.ch

[1] University of Zurich, Zurich, Switzerland

[2] Technische Universität München, Garching, Germany

[3] FHNW, Windisch, Switzerland

Requirements Engineering (Springer)
DOI 10.1007/s00766-017-0274-x Published online 15 July 2017
Preprint in final layout
Final publication is available at https://link.springer.com/article/10.1007/s00766-017-0274-x

[22]. We report on the results of an exploratory study in which we collected and analyzed 10,986,494 tweets about 30 popular software applications in order to understand their relevance for requirements engineering and software evolution. In our study, we investigated general characteristics of tweets (e.g., length, frequency and popularity) and used descriptive statistics to report on the results. Furthermore, we randomly selected 1000 tweets of the collected dataset and manually analyzed them using content analysis techniques [45]. Finally, we investigated the automatic analysis potential by applying machine learning and lexical sentiment analysis on the manually analyzed data for automatically extracting information about their relevance, authors and sentiment.

Our results show that tweets about software applications contain relevant information for software companies. However, due to the large number and high frequency in which tweets are communicated, manual analysis is a cumbersome and time-consuming task. Thus, automated approaches are needed for automatically analyzing tweets about software applications. Furthermore, our results reveal that automated approaches, such as machine learning, are promising for filtering tweets relevant to non-technical stakeholders within the company and the general public, whereas they are less encouraging for detecting tweets that are relevant to technical stakeholders. Additionally, the use of machine learning for detecting tweets about software applications authored by humans or bots yields very promising results. There is a strong positive correlation between the results of automatic lexical sentiment analysis and human judgment when analyzing tweets about software applications that are written by humans.

Automatic approaches can help to identify irrelevant tweets and those authored by bots—which might not need a response from the software company. Additionally, the automatic extraction of sentiment information can be useful when prioritizing tweets, as the sentiment could be an indicator of the urgency of reacting to a specific tweet.

The contributions of this work are threefold. First, our results provide insights into how Twitter is used to communicate about software. Second, we report on the relevance of these tweets on software applications for different stakeholders within software companies. Finally, we show that the application of automated analysis techniques on tweets about software applications is a promising direction for extracting information about their relevance for different stakeholders, authors and sentiment polarity.

The main extensions of this manuscript compared to our previous paper [22] are:

- An experiment to evaluate the potential of using machine learning techniques for the detection of bots-generating tweets about software applications;
- An experiment to evaluate the potential of applying lexical sentiment analysis for the automatic extraction of the sentiments present in tweets about software applications;
- A manual analysis of an additional sample of 1000 tweets using content analysis techniques [45] to study characteristics of bots-generating tweets about software applications. A further analysis of the bot-authored tweets identified by using descriptive statistics;
- An extension of the relevance experiment by analyzing the performance of additional machine learning classifiers;
- A more detailed discussion about the usage and content studies conducted, as well as related work.

The remainder of the paper is structured as follows: Sect. 2 describes the research design. Section 3 describes a study investigating Twitter usage when writing about software applications, while Sect. 4 describes a study analyzing the content of tweets about software applications. Section 5 describes an experiment using machine learning techniques for classifying tweets with respect to their relevance to different stakeholder groups, whereas Sect. 6 describes an experiment also using machine learning to classify the authors of tweets as humans or bots. Section 8 describes an experiment employing lexical sentiment analysis for automatically extracting the sentiment polarity present in tweets. Moreover, Section 9 discusses our main findings, the main threats to validity of our work and sketches future research directions. Finally, Sect. 9 discusses related work and Sect. 10 concludes the paper.

2 Research design

2.1 Research goal and questions

The goal of this study is to explore the current use of Twitter to communicate about software applications and the relevance of this communication for requirements engineering and software evolution. To achieve this goal, we explored the **usage** and **content** of tweets relevant to software applications. Additionally, we investigated the **automation potential** regarding the classification of tweets.

Usage describes how users communicate through Twitter about software applications. In particular, we answered the following question:

- *General characteristics* What are the relevant characteristics of tweets about software applications in terms of frequency, length, interaction, popularity, hashtags and duplication? Which clients are used for posting

them and how often do software companies tweet about their software?

Content describes the different semantic categories present in tweets and their characteristics. In particular, we answered the following questions:

- *Categories* What type of content is present in tweets related to software applications?
- *Relevance* Is the content relevant to software application stakeholders?
- *Sentiment* What is the attitude of users when writing about specific content?

Automation potential describes the potential of applying automation techniques to process tweet content related to software applications. In particular, we answered the following questions:

- *Relevance* What is the performance of supervised machine learning techniques when classifying tweets related to software applications according to its relevance for different stakeholders?
- *Human or Bot* What is the performance of supervised machine learning techniques when classifying tweets as human or bot generated?
- *Sentiment* What is the performance of lexical sentiment analysis when extracting the sentiment polarity present in tweets about software applications?

We chose to focus on filtering irrelevant tweets, detecting tweets authored by bots and extracting sentiment polarities from tweets due to their usage potential during software evolution and requirements engineering. Filtering irrelevant tweets for specific stakeholders and detecting those authored by bots can reduce information overload as it helps to single out those tweets that need a reaction from the software company. Additionally, the automatic extraction of sentiment polarity can help detect the satisfaction level of users tweeting about a specific software application and this information can be useful when prioritizing tweets for software evolution tasks.

2.2 Dataset

We limited the scope of our study to popular mobile and desktop applications. We identified these applications by investigating their number of downloads through charts published by three different distribution platforms.[1]

We collected tweets for the top ten applications of each distribution platform as we assumed they would be mentioned in a large number of tweets. We imported tweets over a period of 2 months starting on November 19, 2015. For this purpose, we employed an open-source library[2] which provides access to the Twitter Search API.[3] This API searches for public tweets published in the past 7–9 days and returns the tweets matching a specified search query. We defined the search query to return tweets that were written in English and whose content included the name of at least one of the 30 chosen applications.[4] The collected tweets can be stand-alone statements written by Twitter users or replies to tweets written by other users. In total, we obtained 10,986,494 tweets about 30 different desktop and mobile software applications, which we used as a dataset for our research.

Table 1 shows an overview of the selected software applications and presents their name, version, domain and the number of imported tweets. For all but four applications, we were able to collect over 1000 tweets. The domains of the applications vary significantly, and we could identify 14 different domains. Included are two systems belonging to the operating system domain. However, in the remainder of the paper we use the term software application to refer to these systems.

2.3 Method

We studied Twitter **usage** to communicate about software applications with the help of descriptive statistics. In particular, we analyzed the *general characteristics* of tweets about software applications. We used content analysis techniques [45] on a random sample of our dataset to study the **content** of tweets about software applications. While conducting this analysis, we manually identified the content *categories* of these tweets and assessed their *relevance* to different stakeholder groups. In addition, we also studied their *sentiment*. Furthermore, we investigated the **automation potential** of tweet analysis by applying machine learning and lexical sentiment analysis techniques on the manually analyzed data. In particular, we measured the performance of the techniques for identifying tweets that are *relevant* to different stakeholders, for filtering tweets about software applications that are generated by *humans* or *bots* and for automatically extracting the *sentiment* polarity present in the tweets.

Table 2 shows the amount of tweets analyzed for the different parts of our study and the techniques used in each one.

[1] http://www.apple.com/itunes/charts/, https://play.google.com/store/apps/top, http://www.amazon.com/best-sellers-software/zgbs/software.

[2] http://www.tweepy.org/.

[3] https://dev.Twitter.com/rest/public/search.

[4] The search query is: tweepy.Cursor (api.search, q = 'APP_NAME -filter:retweets', lang = 'en'), where APP_NAME is the name of the software application. We ran the query iteratively (every 7–9 days) for each software application in our dataset.

Table 1 Dataset

Software	Version*	Domain	#Tweets
Adobe photoshop	Desktop and mobile	Photograph and video	32,663
Afterlight	Mobile	Photograph and video	8734
Akinator the genie	Desktop and mobile	Entertainment	1905
Amazon music	Desktop and mobile	Music and audio	135,042
Amazon shopping	Mobile	Shopping	77,090
Architecture of radio	Mobile	Education	1137
Avast	Desktop and mobile	Security	26,376
Facebook	Desktop and mobile	Social networking	1,917,568
Facetune	Mobile	Photograph and video	2644
Google photos	Desktop and mobile	Photograph and video	74,218
HotSchedules	Mobile	Productivity	3501
Instagram	Desktop and mobile	Photograph and video	1,611,882
Kindle	Desktop and mobile	Books	91,683
LEO privacy guard	Mobile	Tools	411
McAfee	Desktop	Security	34,911
Messenger by facebook	Desktop and mobile	Social networking	75,115
Microsoft office	Desktop and mobile	Productivity	56,501
Norton	Desktop	Security	156,711
Pandora radio	Desktop and mobile	Music and audio	59,869
Snapchat	Mobile	Photograph and video	2,888,469
Spotify music	Desktop and mobile	Music and audio	352,265
True skate	Mobile	Sports	34,765
TurboTax	Desktop and mobile	Finance	14,899
Ultimate guitar tabs	Desktop and mobile	Music and audio	705
Unified remote full	Mobile	Tools	249
Videoshop	Mobile	Photograph and video	2249
WiFi tether router	Mobile	Communication	8
Windows 7	Desktop	Operating system	158,290
Windows 10	Desktop	Operating system	538,655
YouTube	Desktop and mobile	Photograph and video	2,627,979
		Total=	10,986,494

We considered them to have both desktop and mobile versions

* Some software applications have mobile and Web versions (e.g., Facebook and Instagram)

Table 2 Research method overview

Study focus	Used method	Dataset size	Location
Usage	Descriptive statistics	6,437,286	Sect. 3
Content	Content analysis	1000*	Sect. 4
Automation potential: relevance	Machine learning	1000*	Sect. 5
Automation potential: human or bot	Content analysis and machine learning	1000	Sect. 6
Automation potential: sentiment	Lexical sentiment analysis	1000*	Sect. 7

* The same annotated sample, result of the manual content analysis detailed in Sect. 4

3 Usage

In the following, we use descriptive statistics to summarize our findings on how Twitter users communicate about software applications.

3.1 Procedure

During the collection of our dataset, we gathered all tweets which mentioned at least one of the names of the 30 software applications under analysis. However, it is possible that tweets

including the software names are unrelated to the software, lack a clear context or contain a large amount of noise. To focus our work on tweets actually discussing the software application, we used the results of the manual tweet analysis (see Sect. 4) as a filter. We decided to only include software applications in our work in which at least 70% of the tweets included in the manual analysis were actually related to the specific software application and had a clear meaning and context.[5]

Following this strategy, tweets discussing Afterlight, Google Photos, Instagram, McAfee, Norton, True Skate and YouTube were excluded. Furthermore, we also excluded the WiFi Tether Router application from our analysis due to the small number of concerning tweets present in the dataset. In total, 22 applications represented by 6,437,286 tweets were included in our usage analysis.

3.2 Results

Tweets were generated daily for most of the 22 applications. We calculated an average generation **frequency** of 31,336.17 tweets per day, per application (median = 719.06, SD = 11,496.78). However, we also found that this number varies greatly: from 4.02 to 46,588.21 tweets per day (see Fig. 1). These results show that in the long run, for the majority of the analyzed software applications, the number of tweets received per day is too large for manual analysis and filtering.

The average **length** of tweets mentioning software applications was 13.52 words (median = 13, SD = 6.39) or 83.41 characters (median = 81, SD = 36.76).[6] This result shows that the length of tweets mentioning software applications is similar to other tweets (average length of 15.40 words or 86.30 characters) [31]. In their tweets, users can include media (i.e., photographs and videos) and links to enrich the tweets content, which in terms of length are limited to 140 characters. We found that 15.1% of the tweets include links, 4.94% include media, and 4.61% include both.

Twitter allows for bidirectional communication, and users can therefore *reply* to each other and complement their tweets in case clarifications are needed. Reply tweets compose 22.77% of the tweets in our dataset, indicating a high **interaction** between users communicating about software applications.

Users can also react to tweets by *liking* them, which is used to show appreciation for a tweet. Furthermore, they can *re-tweet* a message to their followers. Both actions can be considered as indicators for the **popularity** of a tweet.[7] We found that 34.01% of the tweets in our dataset were liked by other users, with an average of 5.06 likes (median = 1, SD = 138.34). Another 12.06% of the analyzed tweets were re-tweeted, for an average of 5.13 re-tweets per tweet (median = 1, SD = 114.57). Compared with the re-tweeting of random public tweets [63] (2.19% of tweets are re-tweeted), the proportion of re-tweeted tweets about software applications is considerably higher.

Twitter users can include **hashtags** in their tweets, which are keywords preceded with a # character to facilitate grouping and retrieving tweets discussing similar topics. In our dataset, 14.77% of the tweets contain hashtags with an average of 0.28 hashtags per tweet (median = 0, SD = 0.87). Interestingly, only 3.83% of all tweets include the name of the software application as a hashtag.[8] Suh et al. [63] found that including URLs and hashtags in a tweet has a strong correlation with the tweet re-tweetability. However, we found no such correlation in our dataset with only 13.42% of the re-tweeted tweets including hashtags, 7.91% including links and 3.9% including both. Furthermore, the number of hashtags used in the tweet has no effect on the number of re-tweets nor likes the tweet receives (ρ =0.013 and ρ =0.1, respectively).

With respect to company involvement, 21 out of the 22 **companies** developing the software applications had official Twitter accounts dedicated to the analyzed applications[9] and 17 actively used these accounts. Figure 2 presents more details by showing the number of tweets generated by the software companies and their reply rates. Although most software companies are involved in the discussion of their software applications, less than 1% of the total tweets are generated by them. On average, software companies communicate 3.71 tweets per day (median = 0.15, SD = 11.51). This result shows that that the majority of the companies represented in our study use Twitter to communicate with their users, but the frequency of this communication varies.

Another finding of our analysis shows that Twitter users make use of many different **clients** (e.g., Twitter for iPhone, Instagram, Facebook, Twitterfeed, Twitter Web Client and TweetDeck) to communicate their messages. On average, they use 2994.47 different clients (median = 550, SD = 1512.53) per software application. A possible reason for this large variety is that users tend to post tweets in their

[5] We consider tweets that do not belong to the *unrelated*, *unclear* or *noise* categories as fulfilling these criteria. A definition of each category can be found in Table 3.

[6] We follow Twitter's suit and count each link as 23 characters and do not consider media and photographs for the count.

[7] Previous research [35] found that 75% of the re-tweets occur less than a day after the concerned tweet has been posted. Thus, we consider that the number of re-tweets and likes is in most cases complete due to them being collected after the tweet has been present for at most 7–9 days (because of the Twitter API restrictions described in Sect. 2.2).

[8] This includes all different hashtag combinations of the software application name. For example, for Adobe Photoshop: #AdobePhotoshop, #Adobe #Photoshop, etc.

[9] Windows 7 and Windows 10 share the common Twitter account *@Windows*.

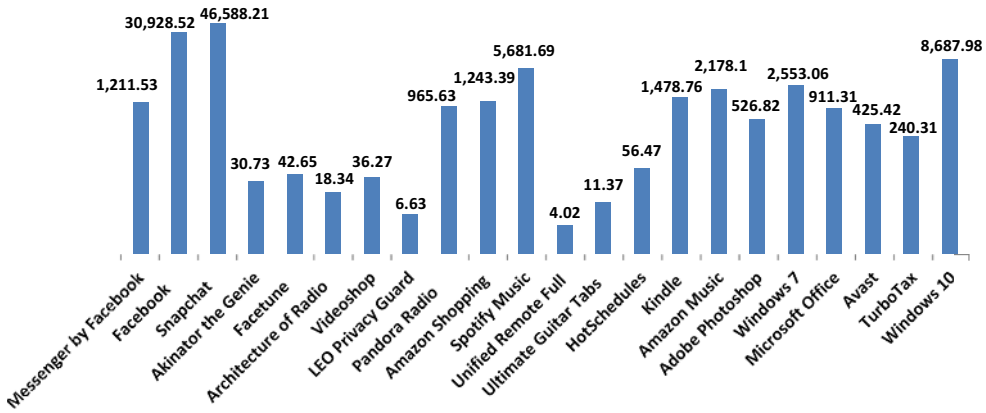

Fig. 1 Daily tweet rate per software application (*graph* shown in logarithmic scale)

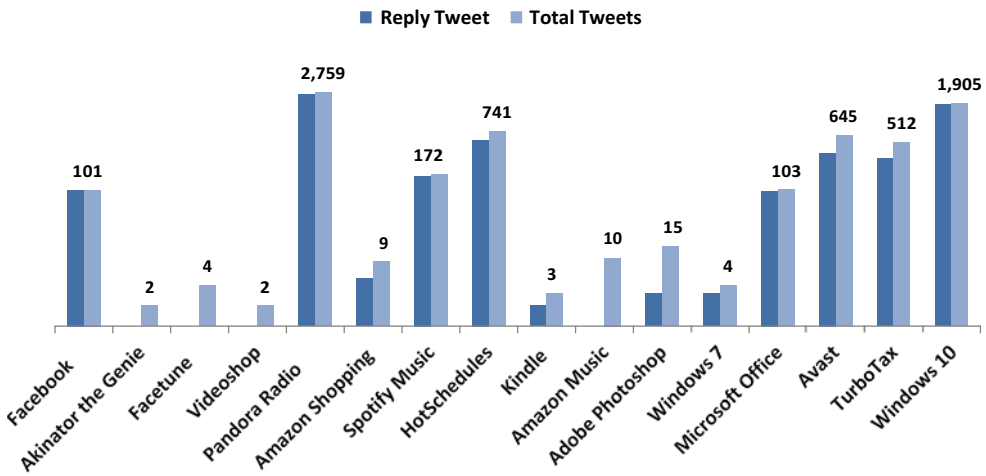

Fig. 2 Total and reply tweets from the software companies, active on Twitter, to which the analyzed software applications belong (*graph* shown in logarithmic scale)

current context [55] and therefore use varying clients, depending on the context from which they tweet.

We also inspected our dataset for **duplicate tweets** (i.e., tweets having exactly the same text, including URLs and hashtags used within the text) and found that 9.5% of the tweets are duplicates (not including re-tweets). In terms of total numbers, the average number of duplicate tweets per software application is 192,207.46 (median = 2639.5, SD = 71,452.93) (see Fig. 3).

In some cases, duplicate tweets might have the same tweet text but different URLs. For example, it is common among spammers on Twitter to take advantage of the URL shortening services to disseminate various URLs that redirect users to malicious Web sites [67]. With this in mind, we also inspected duplicate tweets that have the same text but different URLs. In the analyzed tweets, 17.09% of the tweets are duplicates (but might have different URLs) with an average of 1,183,630.89 duplicate tweets per software application (median = 15,333, SD = 82,217.77). Another Twitter feature that has been abused by spammers is hashtags. Spammers post a large number of hashtags of trending topics which are unrelated to the tweet text to diffuse their tweets [7]. By also removing hashtags in calculating duplicates, the percentage of duplicate tweets increased to 19.02% with an average of 1,143,312.63 duplicate tweets per software application (median = 16,864.5, SD = 90,250.7).

One possible cause for tweet duplication is **bots**. Bots are programs that post tweets automatically to, e.g., lure users into purchases. To further explore the relationship between tweets about software applications and bots, we inspected the users generating the highest numbers of duplicate tweets per software application. For 17 of the 22 software applications, the users producing the majority of the duplicates were actually bots—i.e., Twitter users displaying typical bot communication behavior as described by Chu et al. [13]. For the remaining five applications, the cause for the high number of duplicate tweets was the actual software companies tweeting about their software. Additionally, we found that an average of 154.21 different clients per software application (median = 18, SD = 63.66) includes the term "bot" in their name. This inclusion could

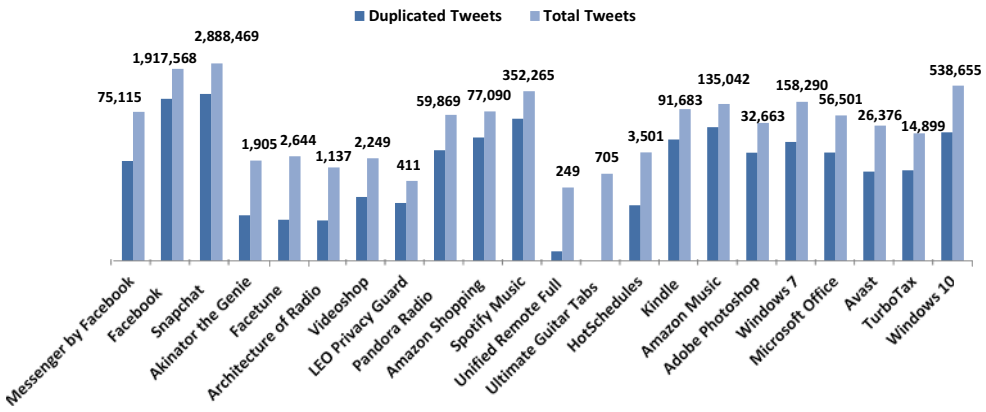

Fig. 3 Total and duplicate tweets per software application (*graph* shown in logarithmic scale)

be a reflection of the clients' actual purpose. The profusion of bots and their association with different client names could also explain the high number of clients per software application found in this study. Bot characteristics are described in more detail in Sect. 6.

4 Content

For the manual tweet analysis, we applied the content analysis method proposed by Neuendorf [45]. During the manual analysis, three annotators, the authors of this paper, systematically analyzed the content of a tweet sample according to an annotation guide. For each tweet, they independently assessed its type of *content*, *relevance* for different stakeholder groups and *sentiment*. We subsequently detail the analysis procedure and describe the results.

4.1 Procedure

The content analysis consisted of the following five steps:

4.1.1 Stakeholder group and content category definition

In this first step, we defined a list of possible content categories present in tweets. Furthermore, we identified different stakeholder groups for whom tweets discussing a particular content could be relevant.

Previous research has analyzed tweets from a technical perspective and investigated how software engineers use Twitter [10] for their development tasks. Our goal was to look at tweets from a more general perspective that is not necessarily technical. Therefore, we used categories found in app reviews [49], which have a more general focus and describe a broader type of content, as a starting point for identifying tweet content categories.

Annotators extended the list by adding new categories found when individually examining the content of 450 tweets from all software applications in our dataset. For each newly defined category, they provided a description and a relevant example. Updates and changes made to the category definitions were visible to all annotators in real time. At the end of this step, similar categories were merged and their definitions were adapted accordingly. The result of this step is a list of 22 categories[10] (see Table 3) which reflects the content of tweets about software.

Based on the results of this content analysis and our general knowledge on software engineering and information needs in software companies, we identified three high-level stakeholder groups for whom the defined content categories could be relevant:

Technical Stakeholders who have a strong and direct participation in the development and evolution of a software application. Examples of such stakeholders include requirements engineers, product owners, project managers, developers and testers.

Non-technical Stakeholders whose work influences or is influenced by software engineering activities. Examples of such stakeholders include people from sales, marketing, legal and human resources departments; they all have a loose participation in the development and evolution of a software application.

General public End-users and potential end-users of a software application who, apart from using the software application, have no link to the software companies.

We discussed our definitions with six requirements engineering experts—mentioned in the Acknowledgements section of this manuscript. They agreed with our view and argued that our classification is well aligned with their understanding of roles and responsibilities within software companies. However, within these discussions, we also conceded that although our classification reflects common

[10] We do not count the categories *unrelated*, *unclear*, *noise* and *other* in this final count.

Table 3 Content categories of tweet messages

Category	Definition
Feature shortcoming	Unsatisfying aspect of an existing feature
Feature strength	Satisfying aspect of an existing feature
Feature request	Request for a new feature
Bug report	Report of an error, flaw, failure or fault
Usage scenario	A way to use the software (e.g., recommended way, workaround)
Hardware constraint	Hardware needed to run the software
Software constraint	Software needed to run the software
General praise	General appreciation of the software focusing on the whole software system
General complaint	General dissatisfaction of the software focusing on the whole software system
Advertisement	Promotion of or suggestion to buy the software
Dissuasion	Advise against the acquisition of the software
Question	Question directly related to the software
How to	Explanation to other users how to use the software
Feature information	Description of a specific feature without any objective evaluation
Software price	Discussion of the price of the software
Compliance issue	Dispute over certain terms of agreement or regulations
Software extension	Description of (planned) extensions of the software
Other product	Reference to another software product
Service	Comment on the service provided by the software
Social interaction	Description of social/personal issues that arise from using the software (i.e., a software feature)
Content related	Comment about content that was created or is available through the software
Job advertisement	Advertisement of a job available in the company developing the software
Noise	Tweet not written in English or containing too many illegible symbols to be understandable
Unclear	Tweet written in English, but the meaning of the tweet is ambiguous or unclear
Unrelated	Tweet not related to the specific software at all
Other	Tweet relevant to the study, but not covered by existing categories

practice, responsibilities and roles might differ from company to company and personal skills and competences might also have an impact on the information needs of stakeholders within software companies.

4.1.2 Annotation guide design

We systematized the manual analysis by creating a guide including definitions and examples of the content categories and sentiment polarity scales, as well as definitions of the different stakeholder groups. Furthermore, we avoided strong disagreements by conducting three annotation trials of 50 tweets each: This led to slight modifications and adjustments of the content categories and sentiment polarity definitions and examples.

Note that within the guide, we did not define the relevance of the content categories for the identified stakeholder groups. In fact, it was an aim of our study to derive this mapping based on the actual tweet content and its context. However, before we started the analysis, we discussed the tweet content that could potentially be relevant to the different stakeholder groups with each other and also with the same group of requirements and software engineering experts, who had already supported us in validating our stakeholder group definitions (see Sect. 4.1.1).

4.1.3 Tweet sampling

We used stratified random sampling to select 1000 tweets for our manual analysis. In total, we selected 33 to 34 tweets per each of the 30 software applications in our dataset. The sample size is comparable to other studies performing manual content analysis of software user content [49, 51].

4.1.4 Tweet sample annotation

In this step, each annotator inspected each of the sampled tweets and labeled it according to the annotation guide. To make this task more efficient and less error-prone, we developed a specialized Web tool for the annotation task. The tool showed the name of the software application,

Table 4 Examples of manual content analysis

Tweet	Categories	Relevance
I'm glad @HotSchedules is offline but I kind of need to know if my shift got approved or not???	Bug report	All stakeholders
Facetune An app to make you good looking.. #Selfies #Photos #Beauty	Advertisement	Non-technical and general public
2000's hip hop radio on pandora	Content related	None
It makes me extremely uncomfortable when people i don't know poke me on facebook	Feature shortcoming and social interactions	All stakeholders
Surface Pro, which is fine. Just a bit buggy. I'd love a real portable alternative. Wish Adobe would sort out their Photoshop app 2/2	Feature request and hardware constraint and other product	All stakeholders

name of the user who wrote the tweet and the tweet itself (including clickable links).

Annotators labeled the tweets not only by its content, but also by the content provided via these links—as these could give relevant context information. During the annotation process, they determined the content categories of the tweet, its relevance to the different stakeholders and the sentiment polarity present in the tweet. Annotators could label more than one content category for each tweet, as tweets can belong to more than one content category (e.g., a tweet can announce or recommend a software and also mention some of the strengths of its features). Similar to the content category, multiple selections were possible when assessing the stakeholder relevance. Sentiment polarities were assessed with a five-level Likert scale ranging from very positive (+2) to very negative (−2).

On average, the annotators took 10.40 h to label the 1000 tweets. This result confirms the large effort needed to conduct manual analysis of user-generated content [18, 26, 27].

4.1.5 Disagreement handling

As the three authors of this paper acted as annotators, all tweets in the sample were annotated three times. To resolve disagreements between the annotators, we used the majority voting scheme. Regarding the analysis of the relevance and category disagreements, the majority voting results yielded no label in 67 cases. Two of the annotators discussed and resolved these disagreements. We resolved sentiment disagreements by converting the categorical values into numerical values (in the [−2, 2] range) and calculating the median.

4.2 Results

4.2.1 Categories

Table 4 shows examples of tweets and the content categories chosen by the annotators. On average, each tweet was associated with 1.24 categories (SD = 0.46). In other words, out of the 1000 tweets within our sample, 217 were assigned to more than one category. However, there are also categories for which no related tweets could be identified within our sample (*software constraint*, *compliance issue* and *service*) and they are therefore not included in the following discussion.

Most tweets (28.30%) belonged to the *advertisement* category, which includes the announcement and recommendation tweets. The second largest group of tweets (25.10%) discussed *content-related* issues, i.e., content managed or produced by the software. The third largest category (15.10%) was tweets *unrelated* to the software application.

Much smaller was the number of tweets within categories providing more relevant information regarding software and requirements evolution. The category *feature shortcoming* was assigned 1.50% of the tweets, *bug report* 0.90% and *feature request* 0.10%.

Although these percentages are low, the large amount of tweets communicated every day for the applications in our data sample suggests that the numbers of relevant tweets are significant and should therefore be considered by software companies planning the next update of their software applications. Assuming that the discussed category distribution also holds for a larger sample and looking at the total numbers, the average software application within our sample would receive, for example, 282 bug reports, 470 feature requests and 31 reports on feature shortcomings on a daily basis.

4.2.2 Relevance

Regarding the relevance of tweets to the different stakeholder groups, 19.30% of the tweets under analysis are relevant to technical stakeholders, 51.50% are relevant to non-technical stakeholders, and 53.20% are relevant to the general public. We highlight examples of tweets relevant to the different stakeholder groups in Table 4. Furthermore, Table 5 presents the percentage of tweets relevant to each

Table 5 Manual content analysis results

Category	Frequency %	Relevance %			Sentiment	
		Technical	Non-technical	General public	Score	Interpretation
Feature shortcoming	1.50	100.00	93.33	93.33	−1.0	Negative
Feature strength	0.80	100.00	100.00	100.00	1.00	Positive
Feature request	0.10	100.00	100.00	100.00	1.00	Positive
Bug report	0.90	100.00	88.89	88.89	0.00	Neutral
Usage scenario	2.50	84.00	96.00	84.00	0.00	Neutral
Hardware constraint	1.10	27.27	54.55	54.55	0.00	Neutral
Software constraint	0.0	N/A	N/A	N/A	N/A	N/A
General praise	2.80	96.43	100.00	100.00	1.00	Positive
General complaint	1.10	100.00	100.00	100.00	−1.00	Negative
Advertisement	28.30	18.37	98.94	98.94	0.00	Neutral
Dissuasion	0.40	100.00	100.00	100.00	0.00	Neutral
Question	0.30	66.67	100.00	100.00	0.00	Neutral
How to	3.70	94.59	97.30	97.30	0.00	Neutral
Feature information	2.50	76.00	100.00	96.00	0.00	Neutral
Software price	8.40	7.14	100.00	100.00	0.00	Neutral
Compliance issue	0.0	N/A	N/A	N/A	N/A	N/A
Software extension	0.10	100.00	100.00	100.00	1.00	Positive
Other product	5.90	59.32	88.14	88.14	0.00	Neutral
Service	0.00	N/A	N/A	N/A	N/A	N/A
Social interactions	3.60	25.00	55.56	50.00	0.00	Neutral
Content related	25.10	8.37	27.49	37.45	0.00	Neutral
Job advertisement	0.30	0.00	100.00	100.00	0.00	Neutral
Noise	1.00	0.00	0.00	0.00	0.00	Neutral
Unclear	9.70	0.00	0.00	0.00	0.00	Neutral
Unrelated	15.10	0.00	0.00	0.00	0.00	Neutral
Other	8.10	7.41	49.38	44.44	0.00	Neutral

Table 6 Relevance tendencies

Technical, non-technical and general public		Non-technical and general public		None	
Feature shortcoming	General praise	Advertisement	Software price	Hardware constraint	Noise
Feature strength	General complaint	Other product		Social interactions	Other
Feature request	Dissuasion	Job advertisement		Content related	
Bug report	How to	Question		Unrelated	
Usage scenario	Software extension	Feature information		Unclear	

stakeholder group for each content category. We inspected the relevance of each category in more detail by analyzing its relevance tendency for the different stakeholder groups. We hold that a category has the *tendency* to be relevant to a specific stakeholder group when over 80% of the tweets belonging to the category are relevant to the group. Table 6 highlights the content categories that tended to be relevant to the different stakeholder groups.

Ten content categories were relevant to all stakeholder groups. Among these categories are *feature shortcoming*, *feature request*, *bug report* and *software extension* which discuss topics relevant to requirements and software evolution tasks. Furthermore, categories such as *general praise*, *general complaint*, *dissuasion* and *feature strength* were also considered to be relevant to all stakeholders, possibly because they give an idea on user satisfaction. The *usage scenario* category was also considered to be relevant to all stakeholder groups, likely because it highlights how users employ the software.

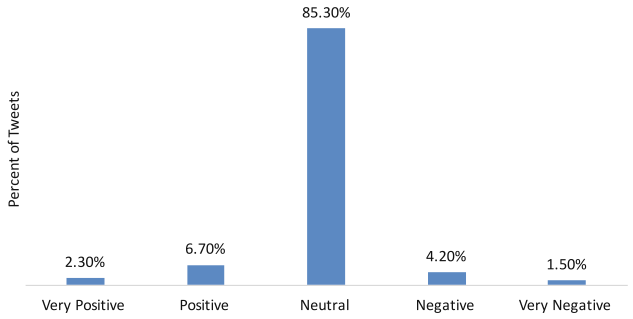

Fig. 4 Sentiment polarity distribution in our dataset (including *noise*, *unclear* and *unrelated* categories)

Six categories were considered to be mainly relevant to non-technical stakeholders and the general public. Among them are the *advertisement*, *other product*, *feature information* and *software price* which are mostly relevant to marketing. The other two categories that were deemed as relevant to non-technical stakeholders and the general public are the *job advertisement* and *question* categories.

The six remaining categories were not considered relevant to any stakeholder group according to our set threshold of 80%. These categories include *social interactions* and *content related*, whose content was in general considered to be relevant only to a small fraction of people and not of interest for the wide general public, as well as the *unclear* and *noise* categories which do not communicate a clear message. Finally, the category *unrelated* was also considered to provide no relevant information for any stakeholder group.

4.2.3 Sentiment

Overall, the analyzed tweets tended to a neutral sentiment with a median score of 0. Figure 4 shows the sentiment polarity distribution in our dataset. This result also remained unchanged when we excluded tweets from categories not related to software applications, such as *noise*, *unclear* and *unrelated*. The large number of tweets with a neutral sentiment could be caused by the proliferation of bot-generated tweets (see Sect. 3). As Table 5 shows, the categories *feature strength*, *feature request*, *general praise* and *software extension* had a positive sentiment polarity with an equal sentiment score median of 1.00. The highest negative sentiment was found in the *feature shortcoming* and *general complaint* categories with a sentiment score median of −1.00.

In general, we consider these sentiment polarities to reflect the nature of the concerned categories: Categories highlighting user satisfaction have a positive sentiment, whereas categories focusing on user dissatisfaction have a negative sentiment. Exceptions are the categories *feature request* and *software extension* with a positive sentiment score median of 1.00. However, only 0.20% (1 tweet per category) of the tweets in our sample were assigned to these categories; hence, we cannot generalize their sentiment.

5 Automation potential: relevance

In this section, we describe an experiment that uses supervised machine learning for classifying tweets according to their *relevance* to the different stakeholder groups identified in our content study (see Sect. 4): technical stakeholders and non-technical stakeholders within the company, as well as the general public. In the following sections, we describe the experiment procedure and its main results.

5.1 Procedure

A tweet can be relevant to different stakeholder groups. For example, as Table 4 shows, the tweet "it makes me extremely uncomfortable when people i don't know poke me on facebook" was considered relevant to all different stakeholder groups, whereas the tweet "Facetune An app to make you good looking.. #Selfies #Photos #Beauty" was regarded as relevant to non-technical stakeholders and the general public.

Multi-label classification refers to the automatic classification of documents, tweets in our case, into one or more labels, relevance categories in this experiment. In our experiment, we used a popular multi-labeling solution, the *binary relevance method* [70]. In this method, a binary classifier is trained for each label and the union operator is applied on the predictions from these independent classifiers, forming the final classification result. We chose this method for our experiment because it is the most simple method for handling multi-label classification and because it is often used as a baseline when solving multi-labeling problems [39].

We trained a classifier for each label using the results of the content analysis reported in Sect. 4.

We compared the performance of five classifiers: Naive Bayes, multinomial Naive Bayes, J48, support vector machines (SVMs) and random forest. The classifier choice was motivated by their good performance when categorizing text [11, 51, 72]. For training and validating the classifiers, we used the manually annotated sample described in Sect. 4.

Our experiment setup consisted of the following three steps:

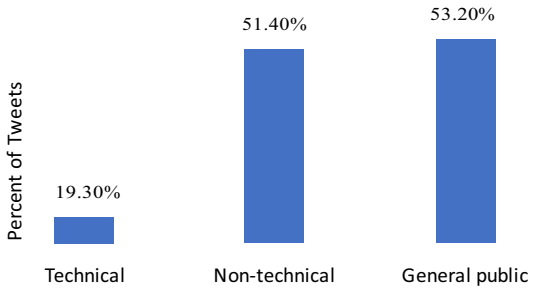

Fig. 5 Relevance distribution among analyzed tweets

5.1.1 Preprocessing

We converted the tweet text to tokens and removed stopwords, i.e., common words that have no specific meaning (e.g., "this", "it", "that"). Additionally, we removed numerical characters and two characters commonly present in tweets: "#" and "@", since we considered that they convey little information about tweet relevance. Lastly, we replaced URLs with a unique marker (i.e., "_Link_") identifying its presence in the tweet text.

5.1.2 Feature weight conversion

We made tweet text understandable to the different classifiers, by converting the text into a vector space model using TF-IDF [44] as a weighting scheme.

5.1.3 Training and evaluation

We trained and evaluated the different classifiers using the results from the content analysis reported in Sect. 4. For this purpose, we exclusively used the text of each tweet, as we considered the tweet text to be the most relevant feature for determining the tweet relevance. The distribution of different relevance categories among the analyzed tweets is shown in Fig. 5.

We performed a tenfold cross-validation for training and evaluating the classifiers.[11] Moreover, we used three metrics traditionally employed in supervised machine learning: precision, recall and F_β-measure for measuring their performance. Their computation is as follows:

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \quad Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{1}$$

TP_i is the number of tweets correctly classified as being relevant to the stakeholder group i, FP_i is the number of tweets incorrectly classified as relevant to stakeholder group i, and FN_i is the number of tweets incorrectly classified as not being relevant to group i. The F_β-measure is defined as follows:

$$F_\beta = (1 + \beta^2)\frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}. \tag{2}$$

In our previous work [22], we used a β value of 1, giving equal importance to precision and recall. However, there is controversy in the requirements engineering community about this manner and some researchers have called for the use of a β value that favors recall over precision when evaluating approaches that automatize requirements engineering tasks by extracting potentially relevant information [8]. In this manuscript, we report on our results from two perspectives and, therefore, consider two β values, F_1 which gives equal importance to precision and recall, as well as F_β which gives recall β times more importance than precision. We followed Berry's recommendation [8] and chose the value of our β empirically, so that:

$$\beta = \frac{t_{TP_i}}{t_{TP_i,FP_i}}, \tag{3}$$

where i designates the technical stakeholders and t_{TP_i} is the average time it takes to determine if a tweet is relevant to technical stakeholders and t_{TP_i,FP_i} is the average time it takes to determine if a single tweet is irrelevant or relevant to any of the groups of stakeholders. We used the time we spent in the creation of the training and testing set, 10.40 h or 624 min (see Sect. 4.1.4) for this computation. Given that 1000 tweets were labeled, $t_{TP_i,FP_i} = \frac{624}{1000} = 0.624$. We compute t_{TP_i} by calculating the ratio of the total time spent on the task to the number of tweets that were found relevant to technical stakeholders, 193 in our case (see Sect. 4.2.2). So, $t_{TP_i} = \frac{624}{193} = 3.23$. We conclude that $\beta = \frac{3.23}{0.624} = 5.18$. We rounded the computed β value to 5 when reporting our results.

We used Meka,[12] a tool specialized in multi-label classification, for the training and evaluation of the relevance classifiers.

5.2 Results

Table 7 presents an overview of the results. Results were comparable when classifying tweets relevant to non-technical stakeholders and the general public. The Naive Bayes classifier had the highest precision, 0.82, whereas multinomial Naive Bayes had the highest recall, 0.82. The random forest classifier had the best F_1-measure, 0.76 for the non-technical classification and 0.77 for the general public classification. Finally, the multinomial Naive Bayes had the best F_5-measure, 0.81. For the classification of tweets relevant to technical stakeholders, the random forest

[11] We performed stratification during our tenfold cross-validation.

[12] http://meka.sourceforge.net, default configuration. Details in "Appendix".

Table 7 Relevance classification results

	Technical				Non-technical				General public			
	Precision	Recall	F_1	F_5	Precision	Recall	F_1	F_5	Precision	Recall	F_1	F_5
Naive Bayes	0.38	0.80	0.52	0.77	0.82	0.67	0.74	0.67	0.82	0.68	0.74	0.68
Multinomial NB	0.30	0.84	0.44	0.79	0.69	0.82	0.75	0.81	0.69	0.82	0.75	0.81
SVM	0.54	0.44	0.48	0.44	0.74	0.77	0.75	0.77	0.74	0.76	0.75	0.76
J48	0.50	0.30	0.38	0.30	0.77	0.73	0.75	0.73	0.78	0.74	0.76	0.74
Random forest	0.73	0.24	0.36	0.25	0.79	0.74	0.76	0.74	0.80	0.74	0.77	0.74

classifier had the highest precision, 0.73, and the multinomial Naive Bayes had the highest recall, 0.84. The Naive Bayes classifier had the highest F_1-measure, 0.52, and the multinomial Naive Bayes had the highest F_5-measure, 0.75.

The precision and recall values for the classification of tweets relevant to non-technical stakeholders and the general public are encouraging as they could be identified with a reasonable precision and recall. The performance similarity for both stakeholder groups could be explained by the fact that a large number of tweets that are relevant to the non-technical stakeholders are also relevant to the general public (see Sect. 4.2.2).

Nevertheless, the results were not as promising for the classification of tweets relevant to technical stakeholders. There was not a single classifier that had reasonable precision and reasonable recall results. Therefore, the application of such classifiers could result in either a considerable loss of relevant information or in the profusion of a large amount of irrelevant information. These results could be a reflection of the lower amount of data that are relevant to the technical stakeholders that were input when training the classifiers (see Fig. 5).

Note that we consider precision and recall values above (or close to) 0.75 to be encouraging. We believe that Twitter can complement other existing requirements elicitation approaches, but not replace them. Because it is not foreseen that companies will use Twitter as their only source of information, recall values do not need to be extremely high. Finally, because it is relatively fast to discard irrelevant information (0.624 min per tweet, see Sect. 5.1.3) we consider that a precision above 0.75 is encouraging. While the classifier will still retrieve noise, it would be fairly fast to discard it and thus, hopefully not discourage stakeholders from using the automated approach.

The choice of which classifier to use in practice is highly dependent on the specific needs of the company and could change based on the context and the received information. For example, software companies receiving a high number of tweets about their application could choose a classifier with a high precision (e.g., random forest) in order to monitor the general mood or potential problems with the application automatically. The random forest classifier has the advantage of more accurately filtering out the noise, and therefore, little additional (potentially manual) filtering would be required, albeit at the cost of loosing a considerable amount of relevant information. Whenever a specific problem becomes apparent requirements engineers and other stakeholders could then use a classifier with a high recall (e.g., Naive Bayes or multinomial Naive Bayes) to obtain most relevant tweets in a given time frame, albeit at the cost of also obtaining some noisy information, which could then be filtered manually or through an additional finer-grained classifier that categorizes into the content categories presented in this work or into a subset of them (see Table 3).

Finally, the binary relevance method has the disadvantage of assuming the independence of each relevance category. Encouraged by the apparent interrelationship between the tweet relevance of non-technical stakeholders and the general public, we compared the binary relevance method against the label powerset method [70], a multi-label classification method that considers each relevance category combination as a single class. The results, however, were comparable to the ones obtained with the binary relevance method.

6 Automation potential: human or bot

This section describes an experiment that uses supervised machine learning to detect if a specific tweet is written by a human or a bot account. Section 3 hypothesized that bots could be a possible reason for the proliferation of duplicate tweets and the large number of different clients. The automatic filtering of tweets associated with bots can help stakeholders within the company to concentrate on tweets generated by humans, in which (contrary to those generated by bots) clarifications, solution strategies and notifications about addressed issues are sometimes needed.

6.1 Procedure

We created a truth set to train and evaluate the different classifiers. In the following sections, we describe its creation process and main characteristics. Additionally, we detail the tweet text preprocessing steps and the training and evaluation of the classifiers.

We evaluated the performance of the same five classifiers described in Sect. 5. However, in this case we performed a single-label classification, as a single tweet can only be generated by a human or a bot, and not by both.

6.1.1 Truth set creation

We created the truth set using Neuendorf's content analysis method [45], previously used in Sect. 4. During the analysis, annotators systematically assessed the content of a tweet sample according to an annotation guide. This analysis was conducted by two authors of this manuscript. For each analyzed tweet, the annotators independently assessed if the user associated with the tweet was a human or a bot. To select additional features, besides the tweet text, to include in the classifier, we analyzed the main characteristics of the bots' tweets by using descriptive statistics. We detail the manual analysis procedure and describe the main characteristics of the generated truth set as follows.

Annotation guide design To systematize our manual analysis, we created an annotation guide which provided descriptions for *human*, *bot* and *undefined* user accounts.

The human and bot characterizations were based on existing work [13]. We included the category undefined for cases in which a decision between human or bot was not possible. This was true when the account of the Twitter user was private or most of the tweets were not in English. A delicate issue is *cyborgs*—accounts that combine machine- and human-generated content. We decided to consider cyborgs accounts in the human category. We argue that all accounts involving human activity, whether they contain machine-generated messages or not, have a "real" end-user or potential user behind the account and thus, should be treated with a higher priority than bots, in which no end-user is involved.

To avoid strong disagreements, we conducted two annotation trials of 20 tweets each. After each trial, the definitions were slightly refined.

Tweet sampling To assure that our training set would be balanced and have enough data points from human and bot accounts, we used stratified random sampling. Our sample had two strata: one with tweets generated from clients whose name contains the word "bot" and the other with tweets generated from clients without the word in their name. Each stratum was equally represented in our sample.

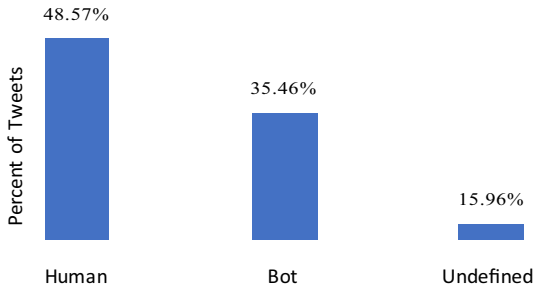

Fig. 6 The distribution of tweets generated by human, bots or undefined users among the analyzed tweets

In total, we sampled 1000 tweets from the 22 software applications analyzed in Sect. 3. Note that this sample is different from the one described in Sect. 4.

Tweet sample annotation The annotators independently labeled each of the 1000 tweets in the sample. The annotation was done through an adapted version of the specialized Web tool used in the content study described in Sect. 4. The tool displayed the name of the software application, name of the user who wrote the tweet (clicking on it redirected to the Twitter profile of the concerned user), the tweet text (including clickable links) and the number of duplicates concerning the tweet in the whole dataset.

During the process, the annotators read the tweet text and went to the account of the concerned user. Annotators were instructed to go through at least 20 tweets and read the user profile description before making their decision of whether the tweet was authored by a human or a bot. Annotators reported 8 h to complete the task.

Truth set characteristics Our human–bot truth set consists of 877 tweets that were marked as human or bot by both annotators, i.e., of the original 1000 annotated tweets, 123 resulted in a disagreement. Figure 6 shows the distribution of tweets generated by the different user categories in our dataset. From the 877 tweets conforming to our truth set, 311 (35%) were unanimously identified as tweeted by bots.

When setting up their Twitter profile, users are required to provide a user name.[13] We found that 10.89% of the users manually identified as bots had the word "bot" in their user name. Another potential source for identifying bots is the software clients used to post the tweet. In our sample, users identified as bots employed 83 distinct software clients (111 distinct clients were used in the whole sample), 42.17% of these clients include the word "bot" in their names, e.g., twittbot.net and Botize. However, these criteria cannot be used exclusively to distinguish bots as

[13] The name that uniquely identifies the user in Twitter; it appears after @ sign (e.g., @twitterapi).

there are software clients used by legitimate users that also include the word in their names, e.g., Tweetbot for Mac.[14]

To analyze how bots communicate with other Twitter users when tweeting about software applications, we inspected their communication patterns. Only 2.57% of the total of tweets identified as being generated by bots were a reply to other users. Moreover, 6.43% of bot tweets were liked by other users with an average of 0.14 likes per tweets. Similarly, a small percentage, 4.50%, of bot tweets were re-tweeted by other users with an average of 0.13 per tweet. These numbers are considerably lower than those found in general tweets about software applications (see Sect. 3.2). A possible interpretation of this observation is that most tweets generated by bots lack intelligent or original content that might attract the attention of other twitter users [13].

A large percent of tweets posted by bots contain links, 77.81%, confirming Chu et al.'s [13] observation that bots usually use links within their tweets to allure users and redirect them to spam or malicious sites.

Furthermore, bot accounts on Twitter tend to include a large number of hashtags in their tweets to reach a wider audience [7]. We found that 40% of the bot-generated tweets in our sample contain hashtags with an average of 0.64 hashtags per tweet (median = 0, SD = 1.25).

Previous research found that posting duplicate tweets is a common behavior in bots [13]. The results of our manual analysis confirm this finding—23.47% of the bot-generated tweets have duplicates with an average of 48.63 duplicates per tweet (median = 18, SD = 85.74 for tweets with at least one duplicate). When considering duplicate tweets with the same tweet text but possibly different enclosed URLs, the percentage of bot-generated tweets having duplicates increased to 67.52%. The average number of duplicates per tweet is 682.58 duplicates (median = 32, SD = 2,008.42 for tweets with at least one duplicate). Likewise, removing hashtags with URLs in calculating duplicates increased the percent of duplicates bot-generated tweets to 72.03% with an average of 644.75 duplicates (median = 35.5, SD = 1,943.13).

Both annotators noted that most of the manually identified bots had either a commercial purpose or were for spreading news or providing entertainment (e.g., quotes, jokes). Furthermore, they noted that many accounts classified as human also contained *machine-generated* messages.

Overall, we can say that the tweets identified as generated by bots from our truth set are strongly characterized by the presence of the word "bot" in their client or user name, by the number of duplicates of the concerned tweets, the presence of links and its low approval and interaction with other users.

[14] http://tapbots.com/tweetbot/mac/.

6.1.2 Preprocessing

We preprocessed the tweet text by following the same preprocessing procedure described in Sect. 5. First, we converted the tweet text to tokens and removed stopwords. Second, we removed numerical characters and tweet-specific characters. Finally, we replaced URLs with unique markers.

6.1.3 Feature weight conversion

We made tweet text understandable to the different classifiers by converting the text into a vector space model using TF-IDF [44] as a weighting scheme.

6.1.4 Training and evaluation

We trained and evaluated our classifiers using the truth set described in Sect. 6.1.1. Besides the tweet text, we input to the classifier two additional features indicating: (1) if the tweet has duplicates in the dataset and (2) if the word "bot" is present in the user or client name. We did not add a feature for the presence of links, as this information is already contained in the tweet text. Other than the user name, we did not add information related to the user account (e.g., number of followers, followees,[15] tweeting time patterns and user profile information) as we were primarily interested in evaluating how the classifier performed when using features directly associated with the tweet and not the user or its account. We performed a tenfold cross-validation for training the classifiers and evaluating our results.[16] For evaluating the accuracy of the classifiers, we used the same metrics from the previous classification experiment.

We used the same method for determining the β of the F_β-measure as in the relevance classification experiment (see Eq. 3). However, in this case the i refers to the tweets authored by humans, t_{TP_i} is the average time it takes to determine if a tweet is authored by humans, and t_{TP_i,FP_i} is the average time it took to annotate each tweet. Given that it took 8 h or 480 min to annotate the whole sample (see Sect. 6.1.1) and that 1000 tweets were annotated, $t_{TP_i,FP_i} = \frac{480}{1000} = 0.48$. We compute t_{TP_i} by calculating the ratio of the total time spent on the annotation task to the number of tweets that were found to be authored by humans, 566 in our case (see Sect. 6.1.1). Thus, $t_{TP_i} = \frac{480}{566} = 0.84$ and $\beta = \frac{0.84}{0.48} = 1.75$. We round this result and, therefore, report on an F_β-measure with $\beta = 2$.

[15] Referred as *friends* by Twitter.

[16] We performed stratification during our tenfold cross-validation.

Table 8 Human–bot classification results

	Human				Bot			
	Precision	Recall	F_5	F_2	Precision	Recall	F_5	F_2
Naive Bayes	0.85	0.86	0.85	0.86	0.81	0.80	0.80	0.80
Multinomial NB	0.88	0.80	0.84	0.81	0.75	0.85	0.80	0.83
SVM	0.89	0.87	0.88	0.87	0.83	0.85	0.84	0.85
J48	0.87	0.88	0.87	0.88	0.83	0.82	0.82	0.82
Random forest	0.85	0.92	0.88	0.91	0.88	0.77	0.82	0.79

The training and evaluation of the classifiers was performed using Weka.[17]

6.2 Results

Table 8 shows an overview of the results. Overall, the results for the human–bot classification are very promising. All classifiers were able to detect humans with a precision and recall above 0.80, with random forest and SVM having the best F_1-measure, 0.88, and random forest having the highest F_2-measure, 0.91. For detecting bots all classifiers had a precision and recall above 0.75, with SVM having the highest F_1-measure and F_2-measure, 0.84 and 0.85, respectively.

Previous work, e.g., [5, 13], achieved very positive results—in the range between 0.80 and 0.95 for both precision and recall for detecting bot accounts. However, a direct comparison with their results is not feasible. Bot account detection takes into account a considerable amount of tweets, opposed to the single tweet consideration we make in this work. Further, bot account detection considers additional characteristics of user accounts over longer time periods, such as tweeting time patterns and following and unfollowing behavior. We were motivated to look into the problem of detecting bots using a more lightweight approach that would not require the software providers to collect a massive amount of information about their (potential) users that is unrelated to the software. Our results show that a reasonable accuracy can be achieved without collecting information that is unrelated to the concerned software application, a fact that could be very appreciated by users with privacy concerns. Future work could focus on the improvement of our results by considering additional attributes that are contained in the text, such as number of hashtags and URLs, length and time of the day it was posted, as done in previous research performing spam detection on single tweets [7, 42]. A direct comparison with existing work in this direction is also unfeasible as the problems have distinct goals. Bot detection focuses on finding messages that are generated automatically, by accounts that are not maintained by humans, whereas spam detection is interested in identifying all messages (or accounts) that have malicious intentions. In our definition, a bot is not necessarily malicious but fully automated.

Human–bot classifiers could help requirements engineers and other stakeholders within the company to identify tweets that might need a response. Additionally, they could be used as a pre-filtering step before classifying tweets into the relevance or content categories presented in this work.

7 Automation potential: sentiment

Sentiment information can be an important measure of user satisfaction and can help prioritize user feedback. In this section, we describe an experiment that uses lexical sentiment analysis to extract the sentiment polarity present in individual tweets.

7.1 Procedure

We use a lexical sentiment analysis tool specialized in short, informal text—SentiStrength [66]—for assigning a numerical score to the sentiment polarity in each tweet. Then, we compare the scores against the content analysis results reported in Sect. 4.

Previous work has shown that SentiStrength has a good performance on Twitter data [65]. It has been used in the software engineering domain for analyzing the sentiment present in user reviews [23, 27], as well as other software engineering artifacts, such as wikis and e-mails [25], commit messages [24, 47] and StackOverflow[18] comments [46].

In the following sections, we describe how SentiStrength works and how we evaluated its results. Note that we did not use machine learning techniques due to the sparsity of positive and negative tweets in the results of our manual content analysis (see Fig. 4), which would not allow for the training of an accurate classifier.

[17] http://www.cs.waikato.ac.nz/ml/weka, default configuration. Details in "Appendix".

[18] http://stackoverflow.com/

Table 9 Sentiment analysis results

Truth set sample	Corr.	Interpretation
All	0.34	Weak
Tweets predicted as not associated with bots	0.36	Weak
Tweets with no URLs	0.47	Moderate
Tweets with no URLs and predicted as not associated with bots	0.60	Strong

7.1.1 Automated lexical sentiment analysis

In SentiStrength, each text snippet (single tweets in our case) is assigned a positive and a negative score. Positive scores are in the [1, 5] range, where 5 denotes an extremely positive sentiment polarity and 1 denotes the absence of positive sentiment polarity. Similarly, negative sentiments range from [−1, −5], where −5 denotes an extremely negative sentiment polarity and −1 indicates the absence of any negative sentiment polarity.

SentiStrength assigns fixed scores to words present in a dictionary where common emoticons are also included. For example, "love" is assigned a score of $\langle 3, -1 \rangle$ and "hate" a $\langle 1, -4 \rangle$ score. Only words present in the dictionary are attributed with a score. Modifier words (e.g., "absolutely," "very") and symbols (e.g., "!", "!!!", "???") can alter the score. For example, "absolutely love" is assigned a score of $\langle 4, -1 \rangle$. The same score is given to "looove" and "love!!!" The sentiment polarity score of a whole tweet is computed by taking the maximum and minimum scores among all the words in the tweet.

We input the text of all analyzed tweets into SentiStrength without any further processing.

7.1.2 Evaluation

We compared the results of SentiStrength to the results from our content study (see Sect. 4). For this purpose, we added the positive and negative scores computed by the tool, obtaining a single sentiment polarity score for each tweet. Following previous work [27, 33], we considered all tweets with a SentiStrength score in the (2, 5] range to be very positive, those in the (1, 2] range as positive and those in the [−1, 1] range as neutral. Tweets with a sentiment polarity score in the [−2, −1) range were considered negative and those with a [−5, −2) range as very negative. We then converted these categorical values into numerical values in the [−2, 2] range, where −2 denotes a very negative sentiment polarity and 2 a very positive sentiment polarity.

We use Spearman's rho correlation for comparing the accuracy of the lexical sentiment analysis tool against the manual sentiment analysis reported in Sect. 4. We employ correlations as they have been previously used to measure the performance of lexical sentiment analysis [15, 27, 65].

7.2 Results

The Spearman's rho correlation coefficient between the automatic sentiment analysis results and the manual annotation is 0.34, indicating a weak positive correlation. Upon further inspection, we learned that lexical sentiment analysis tends to have poor results when analyzing automatically generated tweets, i.e., tweets that are machine generated and can be associated with humans or bots (see entry "Automatic" in column "Generation type" in Table 10). Therefore, we removed the tweets that were associated with bots by our SVM classifier (see Sect. 6) from the truth set as they might be a major source of automatically generated tweets. This modification leads to a weak positive correlation of 0.36. Nevertheless, the sample still contained a large number of automatically generated messages. A possible explanation for this is that many of the users identified as humans are cyborgs and can make heavy use of automatically generated messages. Thus, we filtered out messages that contained URLs, as a manual inspection revealed that most of the automatically generated messages contained them. This modification lead to a moderate positive correlation of 0.47 when considering tweets associated with bots in the evaluation, and of 0.60—a strong positive correlation—when removing messages with links and tweets that were predicted as associated with bots by our classifier. Table 9 summarizes the results, and Table 10 shows examples of human and automatically generated tweets, together with the sentiment polarities assigned in the manual and automated lexical sentiment analysis.

It is important to mention that the lexical sentiment analysis misclassifications also occurred in some tweets written by humans (see Table 9). In these cases, the misclassifications were mostly due to limitations in the lexical sentiment analysis: words not being present in the dictionary, inability to detect context and sarcasm. Part of these limitations could be overcome with machine learning techniques; however, for this purpose a training set that is balanced in terms of the different sentiment polarities is needed.

We conclude that lexical sentiment analysis results are promising on human-generated tweets about software applications. However, it performs poorly on automatically generated tweets about software, and for filtering this

Table 10 Sentiment analysis result examples[1]

Tweet text	Tweet-user classification	Generation type	Manual senti.	Automatic senti.
**tink tink* Hey Ultimate Guitar, bass tabs bloody matter. BASSISTS BLOODY MATTER!!!*	Human	Manual	−1	−1
I JUST EDITED OUT A WATER SPOT USING FACETUNE LOL	Human	Manual	0	1
My Facebook year in review is blank	Human	Manual	0	0
I hate how Facebook tells when your active how I'm gone lurk at these hours	Human	Manual	−2	−2
Has anyone else notices Spotify is killing their battery lately? Used to be fine but now I'm leaking?? all over the place	Human	Manual	−1	−1
I liked a @YouTube video from @androidheadline _link_ Leo Privacy Guard v3 App Review	Human	Automatic	0	1
*Facetune An app to make you good looking.. #Selfies #Photos #Beauty _link_ _link_*	Bot	Automatic	0	1
*Akinator the Genie - Elokence _link_ _link_*	Bot	Automatic	0	0
#TeamFollowBack 113 Million Indians Lost Rs 16,000 on Average to Cyber Crime: Norton #FollowBack	Bot	Automatic	0	−1
*How to use #Microsoft #Office365 to safeguard your business - _link_ _link_*	Human	Automatic	0	1

[1] For readability we replaced URLs with a marker

content, bot classifiers—as the one presented in Sect. 6—are beneficial.

8 Discussion, threats to validity and future work

8.1 Discussion

The key findings of our work are as follows: (1) Tweets include relevant information for requirements engineering and software evolution, (2) automated processing is needed to use this content for informing requirements engineering and software evolution tasks, and (3) automated processing of tweets about software applications to extract information about their relevance, author and sentiment is possible with a reasonable accuracy for some of the studied cases.

We discuss these findings in more detail by revisiting our research questions (see Sect. 2):

Usage Tweets are generated frequently, are short in length and have high popularity. Moreover, there is a considerable number of duplicates among the tweets, possibly due to bots. Additionally, we observed that most of the analyzed companies actively engage in communicating via Twitter about their software applications.

Content Tweets on software applications cover a broad range of categories (see Table 3).

We found that most users employ Twitter to (1) announce or recommend software applications (*advertisement* tweets) and (2) discuss the content available through the applications (*content-related* tweets).

However, we also identified several categories of tweets which are linked to topics relevant to requirements engineering and software evolution, such as *feature shortcomings*, *feature requests*, *bug reports*, *how-tos* and *software extensions*. Although their proportion in the stream of tweets is relatively low, their overall number is significant. These tweets should not be overlooked by software companies and are in particular relevant to technical stakeholders (e.g., requirements engineers). However, such tweets also contain information relevant to non-technical stakeholders within companies as well as the general public.

Overall, the sentiment polarity of tweets about software applications was neutral. However, as expected, we identified positive sentiment polarities for tweets expressing satisfaction (i.e., associated with the *feature strength* and *general praise* categories) and negative sentiment polarities for tweets expressing dissatisfaction (i.e., associated with the *feature shortcoming* and *general complaint* categories).

Our usage and content study shows that Twitter has already established itself as a communication channel between users and stakeholders within software companies where they communicate information relevant to requirements engineering and software evolution. In comparison with other communication channels, such as certain app stores, Twitter has the advantage of allowing bidirectional communication.[19] Bidirectional communication not only enables users to report issues, but also allows stakeholders within the company to ask clarification questions, to inform users about solution strategies and notify them when issues have been addressed. These direct interactions between

[19] The Android and Windows stores allow for bidirectional communication. However, the Apple and Blackberry stores do not.

users and stakeholders within software companies could motivate users to continuously provide high-quality feedback, enabling the evolution of software applications according to user needs.

Automation potential Manually analyzing tweets to identify relevant information for requirements engineering and software evolution can be done in small settings as we did in the present study. However, for the continuous analysis of large quantities of tweets, it is an unsuitable approach. The large number of daily received tweets, the high presence of bot-generated tweets and of tweets that are not relevant to any specific stakeholders (almost 50%) calls for the use of automated analysis techniques.

Our automation potential experiment results are encouraging when classifying tweets that are relevant to non-technical stakeholders and the general public, achieving a F_1-measure of 0.76 and 0.77, respectively. Nevertheless, the results are less encouraging for classifying tweets relevant to technical stakeholders. In this case, the best performing classifier obtained a F_1-measure of 0.52, product of relatively high recall, 0.80—albeit a very low precision, 0.38. None of the analyzed classifiers produced reasonable results for both precision and recall. Therefore, the application of such classifiers could result in a substantial loss of relevant information or in the abundance of large amount of noise. Nevertheless, the classifiers with the highest precision could be used to automatically monitor the general mood or potential issues with the software, at the cost of loosing a considerable amount of relevant information. When these *overview classifiers* detect potential issues, the classifiers with the highest recall could be used to obtain most relevant tweets at a given time frame, albeit while also retrieving a significant amount of noise. The results from the high-recall classifiers could be used as a pre-filtering step before using finer-grained classifiers (or manual analysis) for the additional removal of noisy data.

Additionally, we could classify a tweet as being associated with a human or bot with a F_1-measure of 0.88 and 0.84, respectively. The accurate identification of human and bot users is important in software engineering as it can help identify the users that might need a response to their feedback (i.e., human) from those that do not (i.e., bot).

Finally, our sentiment analysis results show that lexical sentiment analysis is a valuable tool when identifying the sentiment polarity of human-generated content. However, the results are less encouraging when classifying the machine-generated content authored by bots that is also common in some human accounts. Automatic sentiment extraction can help identify pressing issues, causing end-user dissatisfaction, which are more likely to be associated with a negative sentiment polarity.

8.2 Threats to validity

We discuss the main threats to validity of this work subsequently.

Threats to construct validity For the conducted manual analysis studying tweet content, we rely on error-prone human judgement. During this analysis, it was up to each annotator to decide on the content category and stakeholder group assignment for each tweet under analysis. To mitigate the risk of subjective assignments, we involved three annotators in the manual analysis task. Additionally, we created an annotation guide to ensure that all annotators had a shared understanding of the underlying category and stakeholder definitions. To resolve the disagreements, we applied a majority voting scheme. For those tweets in which the majority voting results yielded no label, two annotators discussed and resolved the disagreements.

Threats to internal validity The list of categories used for analyzing tweet content was based on content categories found in a previous study on app reviews [49] and updated using information gained from the manual content analysis of 450 tweets. Nevertheless, considering the vast amount of tweets on software applications, this list could still be incomplete. This represents a threat to internal validity.

Another threat to internal validity is the selection of annotators for conducting the manual tweet analysis. Instead of actual stakeholders representing the identified groups, the authors of this paper conducted the analysis. However, we consider their knowledge regarding requirements and software engineering as sufficient to allow for a meaningful classification. As relevance is highly subjective, we can also expect to see some disagreement even among stakeholders from companies. Regarding the performed analysis, we alleviated this threat by involving requirements and software engineering experts to discuss possible relevance criteria for each stakeholder group.

Threats to external validity We mitigated threats to external validity by selecting software applications from 14 different domains, including mobile and desktop platforms and paid and free apps. Hence, we could obtain insights into the content of tweets on very different software applications. However, we based the selection of our applications on popularity lists from three platforms, and the results might vary for applications available on different platforms or that are less popular. Hence, further studies should be performed to investigate if the results reported in this work hold for applications that are not popular or that are popular in other distribution platforms. Additionally, we did not consider special events occurring in the lifecycle of a software application such as a new release. Such events could potentially have an effect on the

tweets' content and should be investigated in further studies.

We relied on manual content analysis to study the tweets' content, its relevance for different stakeholder groups and the automation potential. A manual analysis on our whole dataset is unfeasible, and for this reason, we used a sample of 1000 tweets. To mitigate generalizability threats, we applied a stratified random sampling strategy, which ensured that tweets about the different software applications—with their category and size diversity—were all analyzed in the same degree. Nevertheless, we only considered a sample of 1000 tweets and further studies with larger samples need to be conducted to conclude if the results reported in this manuscript hold.

Finally, the Twitter API gives access to only a small percent of the publicly posted tweets [1]. To mitigate the threat of dataset incompleteness in our work, we collected a significant sample of tweets (10,986,494 tweets) that were gathered over a period of 2 months.

8.3 Future work

Our exploratory study is a first step toward investigating the use of Twitter to communicate about software applications.

In future work, we will consider additional information about tweets (e.g., length, attached media, number of retweets) to improve the classifiers precision when categorizing by relevance or authorship. Furthermore, the classification results could be improved with the use of ensemble methods [17] that leverage the strengths of different machine learning classifiers.

Another option is to focus on the automated analysis of tweets regarding their content categories (see Table 5). A larger manually annotated dataset could allow for the training of a finer-grained category classifier. Furthermore, a software application-specific classifier could be provided; it could learn about the specific software context and therefore provide higher performance.

Moreover, the approach used in this work for bot detection could be improved by considering additional attributes that are contained in the tweet text, such as number of hashtags and URLs, length and time of the day it was posted, as done in previous research performing spam detection on single tweets [7, 42].

Additionally, the application of machine learning techniques for the sentiment extraction is an interesting direction. Instead of re-inventing the wheel, we recommend an investigation into existing analysis techniques applied on general tweets (see Sect. 9.3) and used in other feedback communication channels, such as app stores, and investigating to what extent they can be applied for analyzing tweets about software applications.

In addition, future research needs to investigate to which degree previous work on mining app reviews (see Sect. 9.1) can be applied on tweets about software applications.

Another interesting line of future work will be to focus on a fine-grained analysis of software applications in certain domains and also certain types of software applications (e.g., mobile vs. desktop applications). Such research might reveal that Twitter is not being used in the same way for all software applications. It can help to understand how certain characteristics of software applications influence their users' communication on Twitter also with respect to the quantity and relevance of these tweets.

During the manual content analysis, we observed that several software companies provide support accounts and that most of the tweets directed to this accounts were highly relevant to technical stakeholders. Therefore, a further investigation of the content addressed to these accounts is an interesting direction.

In general, we consider it to be highly interesting to explore the characteristics of users and their communication within Twitter regarding software applications in more detail (e.g., with the help of social network analysis). Additionally, investigating the use of Twitter while also considering other feedback channels provided by the software companies could help to better understand why and what users communicate about software applications on Twitter.

9 Related work

We discuss three different areas of related work: feedback gathering and analysis, Twitter in the software engineering domain and existing research on Twitter in other disciplines.

9.1 Feedback gathering and analysis

Previous work has highlighted the importance of user feedback to identify ideas for improving the functionality of a software system and its quality [48].

Researchers have started to explore user involvement in requirements and software engineering [32] and have also coined the term crowd-based requirements engineering [20] for describing the idea of letting users contribute to different requirements engineering activities. Work in this field is driven by the rise of social media and mobile applications. For example, previous work has proposed a social network-based approach to identify stakeholders relevant to system development [38], as well as elicit and prioritize requirements [36, 37]. Highly relevant to our work is research on approaches which focus on the elicitation and negotiation of user needs and feedback. In their work on WinBook, Kukreja

and Boehm [34] explore social network functionalities to realize a toolset which supports non-technical stakeholders in gathering and negotiating requirements. Instead of building novel social networks for requirements engineering, Seyff et al. [56] suggested to use existing social network sites and explored the use of Facebook for requirements elicitation, prioritization and negotiation.

In addition to social networks, researchers have also investigated mobile applications to engage users in feedback gathering. For example, Wehrmaker et al. [73] and Seyff et al. [55] provide mobile approaches to continuously elicit user feedback in situ.

Another important stream of research has focused on feedback given by app users via different mobile application distribution platforms. Conducting exploratory studies, Pagano and Maalej [49] and Hoon [29] analyzed the amount, content and rating characteristics of user feedback from mobile application distribution platforms. Again, in line with our findings, the automated processing of feedback is considered to be important. Martin et al. [41] presented a survey in which they investigate the automatic analysis of user feedback present in mobile distribution platforms. In their research, Galvis et al. [18] used a topic modeling algorithm to automatically extract common themes in user feedback that could be useful for requirements engineering and software evolution. Iacob and Harrison [30] extracted feature requests from app store reviews by means of linguistic rules and used a topic modeling algorithm to group the feature requests. Additionally, Gu et al. [21] and Guzman and Maalej [27] presented approaches to extracting features and sentiments from user feedback and Guzman et al. [23] described an approach to retrieving a diverse set of reviews in terms of the software features mentioned in the reviews and the sentiment polarity associated with them. Furthermore, several researchers [26, 40, 51] used machine learning techniques to support the classification of user feedback into categories relevant to software evolution. The automatic analysis of Twitter messages could benefit from the growing work in this area. Moreover, Chen et al. [11], Villarroel et al. [72] and Di Sorbo et al. [16] proposed frameworks for classifying, grouping and ranking user feedback. Palomba et al. [50] presented a framework for linking user reviews to source code changes. We believe that the mining of tweets about software applications could benefit from the results of existing work in the area of app review mining.

9.2 Twitter in software engineering

In our ongoing research, we have started to investigate Twitter as a source of information relevant to requirements engineering and software evolution [22]. The focus of most studies investigating Twitter in the software engineering domain is on developers' use.

Singer et al. [59] conducted a survey and interviews with developers to investigate their use of Twitter. Results report on the difficulty to obtain relevant content due to the information overload caused by the high-frequency generation of tweets; they see the need for automated approaches to analyze tweets. These results are well reflected by our work.

As with our work, content analysis and descriptive statistics were applied in previous studies to describe tweet content on software development. For example, Bougie et al. [10] manually analyzed and grouped the tweets of software developers into different categories. Tian et al. [68] also manually analyzed tweet content focusing on tweets mentioning specific programming languages, libraries and systems and methodologies within their analysis. In a follow-up study, the same dataset was used to investigate the frequency, general characteristics and user interaction among Twitter users [69]. Sharma et al. [58] analyzed a set of tweets containing programming language keywords. In their analysis, they automatically detected popular tweet topics and applied content analysis techniques to further investigate popular topics. Overall, the techniques applied in this previous research are comparable to the work described in this manuscript. However, in our work, we have a different focus and analyze information on software applications from a broader perspective which also includes user requirements and experiences.

Several existing work describes the automated analysis of tweet information. For example, Prasetyo et al. [54] applied machine learning techniques in order to identify tweets mentioning programming languages and containing relevant information for software development. Furthermore, Achananuparp et al. [2] visualized trends within tweets based on aggregated tweet content related to programming language. They use common topics and keywords as a basis for their content analysis. Sharma et al. [57] presented an approach that detects tweets concerning technical issues regarding software development using an unsupervised keyword-based analysis. The automatic processing of tweets about software applications could benefit from the aggregation techniques and classification methods discussed in this section.

9.3 Twitter in other disciplines

The exponential growth of Twitter use has drawn the attention of researchers in different domains. We focus our discussion on related work that investigated the automation potential of different techniques for content classification, bot detection and sentiment analysis on Twitter data with a focus other than software engineering.

The wealth of information shared daily on Twitter makes it a fertile ground for content classification studies. Some of these studies rely only on the tweet content for their classification approaches. For example, Cheng et al. [12] proposed a probabilistic framework for estimating Twitter users' geographical location based purely on the content of their tweets. Yang et al. [75] classified Twitter users according to their interests by applying machine learning techniques on time series generated from their tweets. Other studies leveraged other available information in addition to the tweet content. Sriram et al. [61] used features extracted from user's profile and tweet text to classify tweets into a pre-defined set of categories: News, Events, Opinions, Deals and Private Messages. Hong et al. [28] predicted the popularity of a tweet (using re-tweets as a popularity measure) using a wide spectrum of features based on both user and tweet content information.

The growing popularity of Twitter and its exposed developers APIs have attracted a significant number of automated programs, i.e., bots. In a recent report, Twitter announced that approximately 23 millions of its active users are automated.[20] Many researchers have investigated the detection of bots in Twitter. Chu et al. [13] proposed a classification system to categorize Twitter users as human, bot or cyborg (i.e., bot-assisted humans or human-assisted bots). Davis et al. [14] presented a publicly available service, BotOrNot, for computing a bot-likelihood score for individual Twitter accounts. Both classification systems are based on machine learning techniques that use a combination of features about Twitter users including tweeting behavior, tweet content and account properties for detecting bots among Twitter users.

Another stream of research has focused on detecting spam (malicious bots) on Twitter. For example, considerable research has analyzed spam accounts to understand their behavioral characteristics on Twitter [3, 4, 19, 67]. McCord and Chuah [43] and Singh et al. [60] applied machine learning techniques for detecting spammers on Twitter using users' account information and their recent tweets. In our work, we trained machine learning classifiers to distinguish between tweets generated by bots and those generated by human, based on the tweet text, number of duplicates and the presence of word "bot" in the client name. Future work could investigate if some of the techniques used for spam detection are useful for detecting tweets generated by bots.

Stieglitz and Dang-Xuan [62] found that tweet sentiment is one of the main factors that drive information diffusion in Twitter. A number of researchers investigated how the sentiment expressed in tweets can be used to predict various events. For example, Bollen et al. [9] applied a combination of two lexicon-based tools, OpinionFinder [74] and GPOMS (Google-Profile of Mood States), to analyze tweets sentiment polarity and used it for predicting changes in the stock market. Tumasjan et al. [71] extracted sentiments embedded in political tweets to predict election results, using a lexicon-based analysis tool, LIWC (Linguistic Inquiry and Word Count) [52]. Asur and Huberman [6] applied machine learning techniques to classify sentiment polarity in tweets as positive, negative or neutral and utilized the extracted polarity to improve forecasting of movies revenues. Similar to our work, Thelwall et al. [64] applied SentiStrength to analyze the sentiment polarities associated with popular events in Twitter. Also, Pfitzner et al. [53] analyzed how the sentiment polarity influences the re-tweeting probability of a tweet and used SentiStrength for detecting the sentiment polarity.

We believe that the automatic analysis techniques used on Twitter data from other domains could benefit the early-stage research on Twitter data about software applications.

10 Conclusion

We report on an exploratory study that investigated Twitter *usage* while communicating about software applications, the *content* of tweets about software applications and the *automation potential* of tweet analysis for requirements engineering and software evolution. We found that Twitter is used as a communication channel between users and stakeholders within software companies. Nevertheless, only a small proportion of tweets contains relevant information for technical stakeholders, such as requirements engineers. This finding highlights the need for automated tweet analysis to extract relevant information out of the vast amounts of tweets. Our experiments for classifying tweets according to their relevance to different stakeholders and author types show promising results for some cases. Automated relevance filtering is possible with a F_1-measure ranging from 0.77 to 0.52, while the identification of tweets authored by bots obtained a promising F_1-measure of 0.84. Additionally, the results of an experiment using lexical sentiment analysis for automatically extracting sentiment present in tweets strongly correlate to human judgment when analyzing tweets about software applications authored by humans. We foresee that introducing more sophisticated automated analysis mechanisms will allow software companies to continuously use information stemming from tweets to inform requirements engineering and software evolution.

Acknowledgements We thank Martin Glinz, Dustin Wüest, Melanie Stade, Bernd Brügge, Eya Ben Charrada, Kim Lauenroth, Marjo

[20] http://bit.ly/2kZtPzl.

Kauppinen and Fabiano Dalpiaz for their feedback and discussions. This work was partially supported by the European Commission within the SUPERSEDE project (ID 644018) and a PhD scholarship provided by King Saud University for the second author.

Appendix

Weka configurations

In our work, we applied the default configurations of the classification algorithms as set by Weka.[21]

Naive Bayes

weka.classifiers.bayes.NaiveBayes Class for a Naive Bayes classifier using estimator classes.

- *NumDecimalPlaces* = 2. The number of decimal places to be used for the output of numbers in the model.
- *DoNotCheckCapabilities* = False. If set, the classifier capabilities are not checked before the classifier is built.
- *UsekernelEstimator* = False. If set, kernel density estimator is used rather than normal distribution for numeric attributes.
- *UseSupervisedDiscretization* = False. If set, supervised discretization is to be used to convert numeric attributes to nominal ones.

Multinomial Naive Bayes

weka.classifiers.bayes.NaiveBayesMultinomial: Class for building and using a multinomial Naive Bayes classifier.

- *NumDecimalPlaces* = 2. The number of decimal places to be used for the output of numbers in the model.
- *DoNotCheckCapabilities* = False. If set, the classifier capabilities are not checked before the classifier is built.

SVM

weka.classifiers.functions.SMO: Class that implements John Platt's sequential minimal optimization algorithm for training a support vector classifier.

- *BuildCalibrationModels* = False. This option is used to fit calibration models to the outputs of the support vector machine.
- *C* = 1. The complexity constant.
- *Epsilon* = $1.0e-12$. The epsilon for round-off error.
- *FilterType* = Normalize training data.
- *Kernel* = The kernel to use. weka.classifiers.functions.supportVector.PolyKernel. The polynomial kernel: $K(x, y) = \langle x, y\rangle^p$ *or* $K(x, y) = (\langle x, y\rangle + 1)^p$ with exponent = 1.
- *RandomSeed* = 1. The random number seed for the cross-validation.
- *toleranceParameter* = 0.001. The tolerance parameter.
- *NumDecimalPlaces* = 2. The number of decimal places to be used for the output of numbers in the model.
- *DoNotCheckCapabilities* = False. If set, the classifier capabilities are not checked before the classifier is built.

J48

weka.classifiers.functions.J48: Class for generating a pruned or unpruned C4.5 decision tree.

- *Binarysplits* = False. If set, binary splits are used on nominal attributes when building the trees.
- *CollapseTree* = True. If set, parts are removed that do not reduce training error.
- *ConfidenceFactor* = 0.25. The confidence threshold for pruning.
- *DoNotMakeSplitPointActualValue* = False. If set, the true point is not relocated to an actual data value.
- *minNumObj* = 2. The minimum number of instances per leaf.
- *NumDecimalPlaces* = 2. The number of decimal places to be used for the output of numbers in the model.
- *NumFolds* = 3. The number of folds for reduced error pruning. One fold is used as pruning set.
- *reducedErrorPruning* = False. If set, reduced error pruning is used instead of C.
- *SubtreeRaising* = True. If set, subtree raising is used when pruning.
- *Unpruned* = False. If set, pruning is performed.
- *UseLaplace* = False. If set, Laplace smoothing is used for predicted probabilities.
- *UseMDLcorrection* = True. If set, MDL correction is used when finding splits on numeric attributes.

Random forest

weka.classifiers.trees.RandomForest: Class for constructing a forest of random trees.

- *BagSizePercent* = 100. Size of each bag, as a percentage of the training set size.
- *BreakTiesRandomly* = False. If set, break ties randomly when several attributes look equally good.
- *CalcOutOfBag* = False. Whether to calculate the out-of-bag error.

[21] http://weka.sourceforge.net/doc.dev/overview-summary.html.

- *MaxDepth* = 0. The maximum depth of the tree, 0 for unlimited.
- *NumDecimalPlaces* = 2. The number of decimal places to be used for the output of numbers in the model.
- *NumExecutionSlots* = 1. Number of execution slots.
- *NumFeatures* = 0. The number of features used in random selection.
- *NumIterations* = 100. The number of iterations to be performed.
- *Seed* = 1. The random number seed to be used.

Meka configurations

In our work, we applied the default configurations of the multi-label classification methods as set by Meka.[22] For the base classifiers, we applied the same setting as in Weka.

Binary relevance method

meka.classifiers.multilabel.BR: Class for implementing binary relevance method.

Label powerset method

meka.classifiers.multilabel.LP: Class for implementing label powerset (LP) method.

References

1. Twitter API Rate Limits. https://dev.twitter.com/rest/public/rate-limiting. Accessed 04 April 2017
2. Achananuparp P, Lubis IN, Tian Y, Lo D, Lim EP (2012) Observatory of Trends in Software Related Microblogs. In: Proceedings of the 27th IEEE/ACM international conference on automated software engineering, ASE'12, pp 334–337
3. Almaatouq A, Alabdulkareem A, Nouh M, Shmueli E, Alsaleh M, Singh VK, Alarifi A, Alfaris A, Pentland AS (2014) Twitter: Who Gets Caught? Observed trends in social micro-blogging Spam. In: Proceedings of the 2014 ACM conference on Web Science, WebSci'14, pp 33–41
4. Almaatouq A, Shmueli E, Nouh M, Alabdulkareem A, Singh VK, Alsaleh M, Alarifi A, Alfaris A, Pentland AS (2016) If it looks like a spammer and behaves like a spammer, it must be a spammer: analysis and detection of microblogging spam accounts. Int J Inf Secur 15(5):475–491
5. Alsaleh M, Alarifi A, Al-Salman AM, Alfayez M, Almuhaysin A (2014) TSD: detecting sybil accounts in Twitter. In: Proceedings of the 13th international conference on machine learning and applications, pp 463–469
6. Asur S, Huberman BA (2010) Predicting the future with social media. Proc IEEE/WIC/ACM Int Conf Web Intell Intell Agent Technol 1:492–499
7. Benevenuto F, Magno G, Rodrigues T, Almeida V (2010) Detecting spammers on Twitter. In: Proceedings of collaboration, electronic messaging, anti-abuse and spam conference, CEAS'10
8. Berry DM (2017) Evaluation of tools for hairy requirements engineering and software engineering tasks. In: Technical report, University of Waterloo
9. Bollen J, Mao H, Zeng X (2011) Twitter mood predicts the stock market. J Comput Sci 2(1):1–8
10. Bougie G, Starke J, Storey MA, German DM (2011) Towards understanding Twitter use in software engineering: preliminary findings, ongoing challenges and future questions. In: Proceedings of the 2nd international workshop on Web 2.0 for software engineering, Web2SE'11. ACM, pp 31–36
11. Chen N, Lin J, Hoi SC, Xiao X, Zhang B (2014) AR-miner: mining informative reviews for developers from mobile app marketplace. In: Proceedings of the 36th international conference on software engineering, ICSE'14, pp 767–778
12. Cheng Z, Caverlee J, Lee K (2010) You are where you Tweet: a content-based approach to geo-locating Twitter users. In: Proceedings of the 19th ACM international conference on information and knowledge management, CIKM'10, pp 759–768
13. Chu Z, Gianvecchio S, Wang H, Jajodia S (2012) Detecting automation of Twitter accounts: Are you a human, bot, or cyborg? IEEE Trans Dependable Secure Comput 9(6):811–824
14. Davis CA, Varol O, Ferrara E, Flammini A, Menczer F (2016) BotOrNot: a system to evaluate social bots. In: Proceedings of the 25th international conference companion on world wide web, WWW'16 Companion, pp 273–274
15. De Choudhury M, Counts S (2013) Understanding affect in the workplace via social media. In: Proceedings of the 2013 conference on computer supported cooperative work, CSCW'13, pp 303–316
16. Di Sorbo A, Panichella S, Alexandru CV, Shimagaki J, Visaggio CA, Canfora G, Gall HC (2016) What would users change in my app? Summarizing app reviews for recommending software changes. In: Proceedings of the international symposium on foundations of software engineering, pp 499–510
17. Dietterich IG (2000) Ensemble methods in machine learning. In: Proceedings of the international workshop on multiple classifier systems, pp 1–15
18. Galvis Carreño LV, Winbladh K (2013) Analysis of user comments: an approach for software requirements evolution. In: Proceedings of the 2013 international conference on software engineering, ICSE'13, pp 582–591
19. Grier C, Thomas K, Paxson V, Zhang M (2010) @Spam: the underground on 140 characters or less. In: Proceedings of the 17th ACM conference on computer and communications security, CCS'10, pp 27–37
20. Groen EC, Doerr J, Adam S (2015) Towards crowd-based requirements engineering a research preview. In: Requirements engineering: foundation for software quality. Springer, pp 247–253.
21. Gu X, Kim S (2015) "What Parts of Your Apps are Loved by Users?" (T). In: Proceedings of the 2015 30th IEEE/ACM international conference on automated software engineering, ASE'15, pp 760–770
22. Guzman E, Alkadhi R, Seyff N (2016) A needle in a haystack: What do Twitter users say about software? In: Proceedings of the international conference on requirements engineering, RE'16
23. Guzman E, Aly O, Bruegge B (2015) Retrieving diverse opinions from app reviews. In: Proceedings of the 2015 ACM/IEEE international symposium on empirical software engineering and measurement, ESEM'15, pp 1–10
24. Guzman E, Azócar D, Li Y (2014) Sentiment analysis of commit comments in GitHub: an empirical study. In: Proceedings of the 11th working conference on mining software repositories, MSR'14, pp 352–355. ACM, 2014

[22] http://meka.sourceforge.net/api-1.7/index.html.

25. Guzman E, Bruegge B (2013) Towards emotional awareness in software development teams. In: Proceedings of the 2013 9th joint meeting on foundations of software engineering, ESEC/FSE'13, pp 671–674
26. Guzman E, El-Haliby M, Bruegge B (2015) Ensemble methods for app review classification: an approach for software evolution (N). In: Proceedings of the 2015 30th IEEE/ACM international conference on automated software engineering, ASE'15, pp 771–776
27. Guzman E, Maalej W (2014) How do users like this feature? A fine grained sentiment analysis of app reviews. In: Proceedings of the 2014 IEEE 22nd international requirements engineering conference, RE'14, pp 153–162
28. Hong L, Dan O, Davison BD (2011) Predicting popular messages in Twitter. In: Proceedings of the 20th international conference companion on world wide web, WWW '11, pp 57–58
29. Hoon L, Vasa R, Schneider JG, Grundy J et al (2013) An analysis of the mobile app review landscape: trends and implications. Faculty of information and communication technologies, Swinburne University of Technology, Technical report
30. Iacob C, Harrison R (2013) Retrieving and analyzing mobile apps feature requests from online reviews. In: Proceedings of the working conference on mining software repositories, MSR'13, pp 41–44
31. Jansen BJ, Zhang M, Sobel K, Chowdury A (2009) Twitter power: Tweets as electronic word of mouth. J Am Soc Inf Sci Technol 60(11):2169–2188
32. Johann T, Maalej W (2015) Democratic mass participation of users in requirements engineering? In: Proceedings of the 2015 IEEE 23rd international requirements engineering conference, RE'15, pp 256–261
33. Kucuktunc O, Cambazoglu BB, Weber I, Ferhatosmanoglu H (2012) A large-scale sentiment analysis for Yahoo! answers. In: Proceedings of the 5th ACM international conference on web search and data mining, WSDM '12, pp 633–642
34. Kukreja N, Boehm B (2012) Process implications of social networking-based requirements negotiation tools. In: Proceedings of the international conference on software and system process, ICSSP'12, pp 68–72
35. Kwak H, Lee C, Park H, Moon S (2010) What is Twitter, a social network or a news media? In: Proceedings of the 19th international conference on world wide web, WWW'10, pp 591–600
36. Lim SL, Damian D, Finkelstein A (2011) StakeSource2. 0: using social networks of stakeholders to identify and prioritise requirements. In: Proceedings of the 33rd international conference on software engineering, ICSE'11, pp 1022–1024
37. Lim SL, Finkelstein A (2012) StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation. IEEE Trans Softw Eng 38(3):707–735
38. Lim SL, Quercia D, Finkelstein A (2010) StakeNet: using social networks to analyse the stakeholders of large-scale software projects. In: Proceedings of the 32Nd ACM/IEEE international conference on software engineering, ICSE'10, pp 295–304
39. Luaces O, Díez J, Barranquero J, del Coz JJ, Bahamonde A (2012) Binary relevance efficacy for multilabel classification. Prog Artif Intell 1(4):303–313
40. Maalej W, Nabil H (2015) Bug report, feature request, or simply praise? On automatically classifying app reviews. In: Proceedings of the IEEE 23rd international requirements engineering conference, RE'15, pp 116–125
41. Martin W, Sarro F, Jia Y, Zhang Y, Harman M (2016) A survey of app store analysis for software engineering. In: IEEE transactions on software engineering
42. Martinez-Romo J, Araujo L (2013) Detecting malicious Tweets in trending topics using a statistical analysis of language. Expert Syst Appl 40(8):2992–3000
43. McCord M, Chuah M (2011) Spam detection on Twitter using traditional classifiers. In: Proceedings of the 8th international conference on autonomic and trusted computing, ATC'11, pp 175–186
44. Mitchell TM (1997) Machine Learning, volume 4 of McGraw-Hill Series in Computer Science. McGraw-Hill, New York
45. Neuendorf K (2002) The content analysis guidebook. Sage Publications, Thousand Oaks
46. Novielli N, Calefato F, Lanubile F (2014) Towards discovering the role of emotions in stack overflow. In: Proceedings of the 6th international workshop on social software engineering, SSE'14, pp 33–36
47. Ortu M, Adams B, Destefanis G, Tourani P, Marchesi M, Tonelli R (2015) Are bullies more productive? Empirical study of affectiveness vs. issue fixing time. In: Proceedings of the 12th working conference on mining software repositories, MSR'15, pp 303–313
48. Pagano D, Bruegge B (2013) User involvement in software evolution practice : a case study. In: Proceedings of the international conference on software engineering, ICSE '13, 2013
49. Pagano D, Maalej W (2013) User feedback in the appstore: an empirical study. In: Proceedings of the 21st IEEE international requirements engineering conference, RE'13, pp 125–134
50. Palomba F, Linares-Vásquez M, Bavota G, Oliveto R, Di Penta M, Poshyvanyk D, De Lucia A (2015) User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In: Proceedings of the IEEE international conference on software maintenance and evolution, ICSME'15, pp 291–300
51. Panichella S, Di Sorbo A, Guzman E, Visaggio C, Canfora G, Gall H (2015) How can i improve my app? Classifying user reviews for software maintenance and evolution. In: Proceedings of the 31st international conference on software maintenance and evolution, ICSME'15, pp 281–290
52. Pennebaker J, Chung C, Ireland M (2007) The development and psychological properties of LIWC2007
53. Pfitzner R, Garas A, Schweitzer E (2012) Emotional divergence influences information spreading in Twitter. In: Proceedings of the 6th international AAAI conference on weblogs and social media, ICWSM'12
54. Prasetyo PK, Lo D, Achananuparp P, Tian Y, Lim EP (2012) Automatic classification of software related microblogs. In: Proceedings of the 28th IEEE international conference on software maintenance, ICSM'12, pp 596–599
55. Seyff N, Ollmann G, Bortenschlager M (2014) AppEcho: a user-driven, in situ feedback approach for mobile platforms and applications. In: Proceedings of the 1st international conference on mobile software engineering and systems, MOBILESoft'14, pp 99–108
56. Seyff N, Todoran I, Caluser K, Singer L, Glinz M (2015) Using popular social network sites to support requirements elicitation, prioritization and negotiation. J Internet Serv Appl 6(1):1–16
57. Sharma A, Tian Y, Lo D (2015) NIRMAL: automatic identification of software relevant Tweets leveraging language model. In: Proceedings of the IEEE 22nd international conference on software analysis, evolution and reengineering, SANER'15, pp 449–458
58. Sharma A, Tian Y, Lo D (2015) What's hot in software engineering Twitter space? In: Proceedings of the IEEE international conference on software maintenance and evolution, ICSME'15. IEEE, pp 541–545
59. Singer L, Figueira Filho F, Storey MA (2014) Software engineering at the speed of light: how developers stay current using Twitter. In: Proceedings of the 36th international conference on software engineering, ICSE'14, pp 211–221
60. Singh M, Bansal D, Sofat S (2014) Detecting malicious users in Twitter using classifiers. In: Proceedings of the 7th international

conference on security of information and networks, SIN'14, pp 247–253
61. Sriram B, Fuhry D, Demir E, Ferhatosmanoglu H, Demirbas M (2010) Short text classification in Twitter to improve information filtering. In: Proceedings of the 33rd international ACM SIGIR conference on research and development in information retrieval, SIGIR'10, pp 841–842
62. Stieglitz S, Dang-Xuan L (2013) Emotions and information diffusion in social media—sentiment of microblogs and sharing behavior. J Manag Inf Syst 29(4):217–248
63. Suh B, Hong L, Pirolli P, Chi EH (2010) Want to be retweeted? Large scale analytics on factors impacting retweet in Twitter network. In: Proceedings of the IEEE 2nd international conference on social computing, SocialCom'10, pp 177–184
64. Thelwall M, Buckley K, Paltoglou G (2011) Sentiment in Twitter events. J Am Soc Inf Sci Technol 62(2):406–418
65. Thelwall M, Buckley K, Paltoglou G (2012) Sentiment strength detection for the social web. J Am Soc Inf Sci Technol 63(1):163–173
66. Thelwall M, Buckley K, Paltoglou G, Cai D, Kappas A (2010) Sentiment strength detection in short informal text. J Am Soc Inf Sci Technol 61(12):2544–2558
67. Thomas K, Grier C, Song D, Paxson V (2011) Suspended accounts in retrospect: an analysis of Twitter spam. In: Proceedings of the 2011 ACM SIGCOMM conference on internet measurement conference, IMC'11, pp 243–258
68. Tian Y, Achananuparp P, Lubis IN, Lo D, Lim EP (2012) What does software engineering community microblog about? In: Proceedings of the 9th IEEE working conference on mining software repositories, MSR'12, pp 247–250
69. Tian Y, Lo D (2014) An exploratory study on software microblogger behaviors. In: Proceedings of the IEEE 4th workshop on mining unstructured data, MUD'14. IEEE, pp 1–5
70. Tsoumakas G, Katakis I (2007) Multi-label classification: an overview. Int J Data Warehous Min 3:1–13
71. Tumasjan A, Sprenger T, Sandner P, Welpe I (2010) Predicting elections with Twitter: What 140 characters reveal about political sentiment. In: Proceedings of the 4th international AAAI conference on weblogs and social media, ICWSM'10, pp 178–185
72. Villarroel L, Bavota G, Russo B, Oliveto R, Di Penta M (2016) Release planning of mobile apps based on user reviews. In: Proceedings of the 38th international conference on software engineering, ICSE'16, pp 14–24
73. Wehrmaker T, Gärtner S, Schneider K (2012) Contexter feedback system. In: Proceedings of the 34th international conference on software engineering, ICSE'12, pp 1459–1460
74. Wilson T, Hoffmann P, Somasundaran S, Kessler J, Wiebe J, Choi Y, Cardie C, Riloff E, Patwardhan S (2005) OpinionFinder: a system for subjectivity analysis. In: Proceedings of HLT/EMNLP on interactive demonstrations, HLT-Demo'05, pp 34–35
75. Yang T, Lee D, Yan S (2013) Steeler Nation, 12th man, and boo birds: classifying Twitter user interests using time series. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining, ASONAM'13, pp 684–691