Observatory of Trends in Software Related Microblogs

Palakorn Achananuparp, Ibrahim Nelman Lubis, Yuan Tian, David Lo, Ee-Peng Lim
Singapore Management University, Singapore
palakorna@smu.edu.sg, lubisnelman@smu.edu.sg, yuan.tian.2011@exchange.smu.edu.sg,
davidlo@smu.edu.sg, eplim@smu.edu.sg

***Abstract*—Microblogging has recently become a popular means to disseminate information among millions of people. Interestingly, software developers also use microblog to communicate with one another. In Twitter, many developers microblog or *tweet* about bugs, new IDEs, new programming languages, etc. Different from traditional media, microblog users tend to focus on recency and informality of content. Many tweet contents are relatively more personal and opinionated, compared to that of traditional news report. Thus, by analyzing microblogs, one could get the real-time information about what people are interested in or feel toward a particular topic. In this paper, we describe our microblog observatory that aggregates more than 58,000 Twitter feeds, captures software-related tweets, and computes trends from across topics and time points. Finally, we present the results to the end users via a web interface available at http://research.larc.smu.edu.sg/palanteer/swdev.**

I. Introduction

Microblogging services, such as Twitter, provide a convenient way for millions of people to communicate with one another thanks to their informal and timely natures. Twitter users typically compose a short microblog (up to 140 characters in length), also known as *tweet*, to express their thoughts on various subjects. Because of the sheer sizes and scopes of tweets, Twitter is a great venue for studying information diffusion among the networks of users.

Interestingly, software developers and users of software systems also tweet. One could potentially learn many interesting things from their tweets. Information such as: new features of a software system, new programming methodology, new conferences, new bugs, new issues, new solutions, new feature requests, etc., could be disseminated via the tweets. These provide a rich source of information well suited for software development where many new "events" happen periodically – new releases, new features, new bugs, etc. If these tweets could be distilled into knowledge, one could in effect learn from the wisdom of the crowd. Moreover, given that informal communication plays an important role in software development projects [3], [4], [7], a study of how microblogs facilitate communications in software development activities may help uncover valuable insights into many software development processes.

There have been a number of studies that analyze microblogs in general and Twitter [8]. However, all of these studies focus on general Twitter users. Unfortunately, there has been little study on investigating the behavior of a particular sub-community of Twitter users. In a related study, we manually investigate what types of information content are contained in software engineering community microblogs, e.g., commercials, opinion, tips, etc [11]. In this study, different from the above, we build a visual analytics observatory that computes trends (both across topics and time points). We also investigate the different purposes of tweeting among software developers in another study[11].

In the software engineering front, recently a number of studies analyze social media sources such as: blogs, forums, etc. Pagano and Maalej investigated how software developers and other members of open source communities blog [7]. Gottipati et al. built a search engine to leverage information available in software forums [4]. Guzzi et al., Begel et al., and Treude and Storey proposed the integration of social media into software development processes [5], [2], [12]. However, there has been no study that provides a solution that is able to aggregate and analyze the wealth of software engineering information available in microblogs.

It is challenging to analyze Twitter data. First, information available in Twitter about a particular topics is distributed in many tweets made by many users. Second, the amount of information is gigantic – there are millions of users producing millions of tweets daily. As such, storing *all* tweets is practically infeasible. Third, many Twitter users tweet about various subjects, not necessarily limited to software development, thus filtering out the irrelevant tweets is not trivial. Fourth, the massive number of tweets may contain useful *software-related trends* which are neither apparent nor readily available to Twitter users. Although one can search for tweets mentioning software-related terms using the default interface in Twitter, the search results only contain the recent tweets published in the past week. Furthermore, it is difficult to analyze or make sense of the trends from those results.

To address the above challenges, there is a need for an approach that could aggregate tweets made by various Twitter users, in particular those that are likely to tweet about software development. The approach needs to be scalable enough to handle a large number of tweets. The approach needs to allow users to perform *visual analytics* to better understand recent trends and developments from the mass of data. Furthermore, it should allow for periodic updates as new tweets are made daily. In this study, we present a system that satisfies the above criteria.

We build a visual analytics observatory that capture *topical*

trends and *longitudinal* trends. Topical trends capture relative popularity of various related topics, e.g., relative popularity of various programming languages. Longitudinal trends capture relative popularity of a topic at various points in time, e.g., time periods where people microblog more about Javascript. We believe these trends could provide various insights to developers, e.g., to find important emerging topics that other developers care about, such as an important release, etc.

We took a user-centric approach of gathering the tweet data to be used by such an observatory. The basic idea is to find a balance way of maximizing the number of tweets to be collected and maximizing the relevancy and completeness of the tweet data pertaining to software engineering topics. To that end, we first build a sizable set of candidate users that are more likely to tweet about software engineering related topics. Next, we periodically download tweets made by these users, pre-process such tweets, store, and index them in a database. This database is later processed to compute topical and longitudinal trends which are later presented to the end users via a web interface. The prototype of our web application is accessible via a following URL:

http://research.larc.smu.edu.sg/palanteer/swdev

We have also performed a preliminary study on approximately 58,000 Twitter users for tweets published between June 2011 through November 2011. The contributions of this study are as follows:

1) We propose a solution that aggregates and processes software engineering related microblogs into topical and longitudinal trends and present them to the end users for visual analytics.
2) We have performed a preliminary analysis on several interesting trends that we capture from 58,000 Twitter feeds. We have also implemented our solution as a publicly accessible web application.

The structure of this paper is as follows. In Section II, we present our proposed framework. In Section III, we present our preliminary study and highlight some interesting topical and longitudinal trends. In Section IV, we present related studies. We conclude and describe future work in Section V.

II. Proposed Framework

Our framework is illustrated in Figure 1; It is composed of three blocks: *User Base Creator*, *Tweet Processor*, and *User Interface*. The *User Base Creator* block extracts a set of Twitter users that are likely to post software engineering related contents. The *Tweet Processor* block extracts tweets produced by the selected set of users. It also pre-processes, indexes, and stores the tweets. The *User Interface* block presents an web interface for the users to query and analyze various topics.

A. User Base Creator

Among the millions Twitter users, not everyone tweets about software engineering topics. Thus, we would need to create a sizable user base that is likely to produce software engineering related tweets.

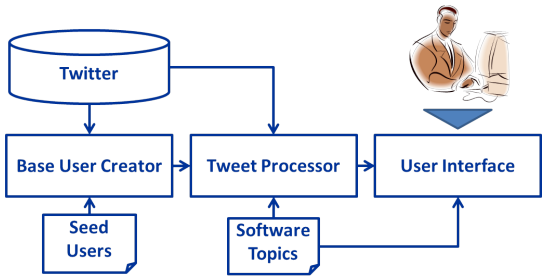

Fig. 1. **Proposed Framework**

To accomplish this, our system takes in a set of seed users ($SEED$). The seed users are well-known Twitter users that actively tweet about software topics. These users are manually identified; for example, *Jeff Atwood*, *Jason Fried*, and *John Resig*. We assume that any Twitter users who follow at least n popular developers are interested in software development and likely to tweet about the topic. Given the assumption, we proceed to expand our software-centric user base by traversing the seed users's friend and follower networks.

We make use of the *follow links* in Twitter. If Bob *follows* Alice in Twitter, any tweets published by Alice will be automatically broadcast to Bob. We could traverse the follow links of these seed users to substantially expand the user base. Our strategy is to obtain the following set of users:

$$UBase = SEED \bigcup \{u | u \ follows \ \geq \ n \ users \ in \ SEED\}$$

By default, we set the value n in the above equation to 5. That is, we include a Twitter user to $UBase$ if she follows at least 5 seed users.

B. Twitter Data Processor

Our Twitter data processor block consists of three steps: tweet and follow links download, tweet pre-processing and indexing, and trend analysis.

Tweet & Follow Link Download. After the initial user base is identified, we automatically download all the latest tweets published by these users using the Twitter REST API and a Twitter whitelisted account. A whitelisted account is permitted to make 20,000 API calls per hour, as opposed to 350 calls per hour of a non-whitelisted one. Because of an API limitation[1], up to 3,200 tweets of any user can be retrieved at a given time. Thus, at the initial tweet download, the completeness of the tweet data are constrained by the API functionality. However, after polling data for the first time, we continue to collect $UBase$'s tweets on a daily basis. This guarantees a near complete snapshot of the subsequent tweet data as no users in $UBase$ publishes more than 3,200 tweets in a single day. In addition, we also download the follow links for all users in $UBase$ everyday using the API. After which, the set of users in $UBase$ are updated according to the newly inserted or

[1]https://dev.twitter.com/docs/rate-limiting

removed friends and followers of $UBase$. On average, 177K tweets and 100K follow links are downloaded in one day.

Tweet Pre-Processing & Indexing. We then perform common text pre-processing steps namely: tokenization, stopword removal, and stemming. We use whitespace and punctuations as the delimiters for tokenization, remove common English language stop words, and utilize Porter stemmer to reduce a word to its root form [9]. We manually mark some technical synonyms and jargons that should not be stemmed e.g., C# vs. CSharp, C++, etc. Finally, we index all processed tweets. To accomplish these tasks, we employ an open source search platform Apache Solr[2]. Our tweet processor can be re-run at various points in time such that new tweets could be included into the repository.

Trend Analysis. Next, we process the tweets to compute both topical and longitudinal trends. To compute topical trend, we manually select a set of 100 software-related topics, e.g., JavaScript, Scrum, etc., from relevant Wikipedia pages and popular StackOverflow.com's tags. We further divide them into three groups namely: 1. *Programming Languages*, 2. *Frameworks, Libraries, and Systems*, and 3. *Programming Concepts and Methodologies*. We then compute for each topic the number of tweets mentioning the topic at a specific time period. Topics that are more frequently mentioned are more popular than others. To compute longitudinal trend of a particular topic or keyword, we compute the number of tweets containing it at various points in time. We thus could compute the popularity of various topics and the popularity of a topic at various time points. Note that although Twitter has officially published a list of trending topics for a specific locale, these topics are extracted from *all tweets*. We believe the Twitter trending topics are not particularly useful to us since they are not categorized and likely dominated by many non-software related topics.

C. User Interface

A snapshot of our user interface is shown in Figure 2. The main user interface shows the four topic groups. For each group, we show *topical trend*; topics that are more frequently tweeted in the repository are shown using a larger text size.

A user can click any topics in the four groups. After this, a line chart showing the number of tweets containing the topic over time would be shown. This chart represents the *longitudinal trend* of the topic. An example of this chart for "JavaScript" is shown in Figure 3. Users could also investigate the actual tweets made at various points in time by clicking at the various points in the line chart. An example of the resultant UI showing actual tweets made at September 1, 2011 is shown in Figure 4. Aside from clicking the topics, one can also enter any free-text queries in the top right text box. A similar line chart would also be plotted for this query. Multiple queries can also be submitted together to generate multiple trend lines for comparison.

[2]http://lucene.apache.org/solr/

Fig. 2. **User Interface & Topical Trends**

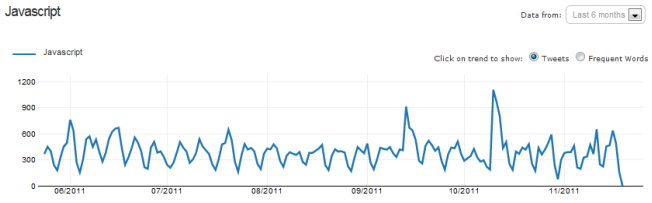

Fig. 3. **Longitudinal Trend on "JavaScript"**

Fig. 4. **Tweets on "JavaScript" made at September 1, 2011**

III. Preliminary Study

We have conducted a preliminary study on a reasonably large number of Twitter feeds. In this section, we briefly describe our dataset followed by some interesting trends that we find in this preliminary study.

A. Dataset

Our framework takes in a number of parameters: a set of s seed users, a set of t topics, and the parameter n that we use to expand the seed users to our user base. For the set of seed users, we utilize the list provided in [1] which identifies top-100 software developers that tweet. As described in Section II, we include 100 software-related topics and divide them into 3 main categories. In this work, the values of s, t, and n are 100, 100, and 5, respectively. As of November 23, 2011, the whole dataset comprises approximately 58,000 unique users, 18 million follow links, and 76 million tweets in total.

B. Interesting Trends

With the topical trend analysis, users can visually inspect the topics in which software engineering community in Twitter

are interested. For example, we find that *JavaScript*, *Ruby*, and *Java* are the three most popular programming languages mentioned by Twitter users during the last 24 hours of November 25, 2011, respectively. The corresponding topical trend interface is shown in Figure 2. Similarly, by comparing different software engineering concepts and methodologies, we find that *Agile*, *Collection*, and *Open Source*, are the three most popular software-related concepts, respectively. Interestingly, since we perform a simple term frequency count, the popularity of terms with multiple meanings like Collection may be grossly overestimated as the counting includes tweets mentioning Collection as a data structure as well as those using it generically, e.g. DVD collection, etc.

With longitudinal trend analysis, uses can capture the popularity of a topic across time points. For example, from Figure 3, we can see that an interest in JavaScript goes up and down periodically. We can also identify events which correspond to the unusual peaks or dips in the trend line. For example, we notice that there is an unusual peak in the number of tweets on October 10, 2011. Upon further inspection, we find that Google unveiled a new programming language *Dart* on that day. Figure 5 shows another longitudinal trend for the Scrum software development methodology. We notice that its popularity is again periodic. The peaks and dips however look much alike and there is not much anomaly.

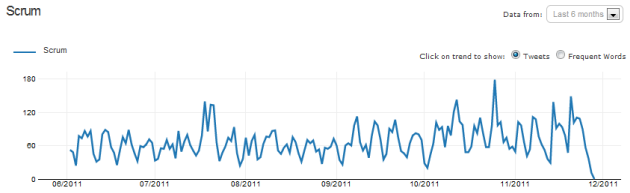

Fig. 5. **Longitudinal Trend on "Scrum"**

IV. Related Work

Social Media for Software Engineering. There have been a number of studies that proposed the integration of social media with IDE and software development [5], [2], [12]. Pagano and Maleej analyzed how open source communities blog [7]. Gottipati et al. built a semantic search engine to effectively find answers in software forums [4]. In this study, we build an analytics engine that downloads, pre-processes, indexes, and stores microblog data. It also computes trends from the data, and presents them to the end users for insights.

Social Network Mining in Software Engineering. There have also been a number of studies in software engineering domain that analyzed socio-technical network and utilized social network mining techniques. Bird et al. analyzed social network created from email communications among developers [3]. Surian et al. and Hong et al. analyzed developer socio-technical network in SourceForge.Net [10], [6].

Twitter Analytics There have been a few Twitter Analytics applications that allow the users to explore trends and insights extracted from the tweet data. In particular, our solution can be contrasted with the Archivist[3]. Unlike the Archivist, which is not domain-specific, our tweet observatory specifically focuses on the software engineering domain. Through domain knowledge, we only keep track of a specific community of users who likely tweet about software-related topics. Moreover, our interface encourages more serendipitous discovery of other interesting software engineering topics through a topical trend browsing. Since we use the Twitter REST API and a whitelisted account to periodically collect the tweet and follow-link data, we are able to capture a much larger volume of tweets and these tweets are also less prone to noise than those collected by the Archivist. For instance, given a programming language topic *Java*, the tweets collected by the Archivist for this query are mostly about "Java, Indonesia" while our system's are mostly about the programming language. As a result, the Archivist's aggregated visualizations may not accurately depict the trend for Java programming language. This is due to the fact that the Archivist includes the tweets from all Twitter users while our system selectively collect the tweets from the relevant users only.

V. Conclusion and Future Work

In this work, we propose a system that captures trends in software related microblogs from Twitter. It collates a pool of Twitter users that are likely to tweet about software development based on a seed set of popular developers. It then collects, processes, and presents the data to the end users as trends via the topical and longitudinal trend analysis interface. The system is implemented as a publicly available web application. We have also found some interesting trends in our preliminary experiments.

In the future, we plan to perform more analysis on what software engineering community microblogs about. We would also like to improve our visual observatory by incorporating additional features. Furthermore, we plan to build a system that could automatically summarize microblog contents, identify important events, and allow users to discover more nuggets of knowledge from the masssive microblog data.

References

[1] http://www.noop.nl/2009/02/twitter-top-100-for-software-developers.html.

[2] A. Begel, R. DeLine, and T. Zimmermann, "Social media for software engineering," in *Workshop on Future of Software Engineering Research*, 2010.

[3] C. Bird, A. Gourley, P. T. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *MSR*, 2006, pp. 137–143.

[4] S. Gottipati, D. Lo, and J. Jiang, "Finding answers in software forums." in *ASE*, 2011.

[5] A. Guzzi, M. Pinzger, and A. van Deursen, "Combining micro-blogging and ide interactions to support developers in their quests." in *ICSM*, 2010.

[6] Q. Hong, S. Kim, S. Cheung, and C. Bird, "Understanding a developer social network and its evolution," in *ICSM*, 2011.

[7] D. Pagano and W. Maalej, "How do developers blog? an exploratory study," in *MSR*, 2011.

[8] B. Poblete, R. Garcia, M. Mendoza, and A. Jaimes, "Do all birds tweet the same?: characterizing twitter around the world," in *CIKM*, 2011.

[9] M. Porter, "An algorithm for suffix stripping," *Program*, 1980.

[10] D. Surian, N. Liu, D. Lo, H. Tong, E.-P. Lim, and C. Faloutsos, "Recommending people in developers' collaboration network," in *WCRE*, 2011.

[11] Y. Tian, P. Achananuparp, I. Lubis, D. Lo, and E.-P. Lim, "What does software engineering community microblog about?" in *Under submission, also available at: http://research.larc.smu.edu.sg/papers/tweet-analysis.pdf*, 2011.

[12] C. Treude and M. Storey, "How tagging helps bridge the gap between social and technical aspects in software development?" in *ICSE*, 2009.

[3] http://archivist.visitmix.com/