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. 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 up-to-date 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 70,000 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.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: Misc.

General Terms

Human factors, Design

Keywords

Software Development, Twitter, Visualization, Exploration

1. 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. 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 [2, 3, 5], a study of how microblogs facilitate communications in software development activities may help uncover valuable insights into many software development processes.

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 [7]. In this study, different from the above, we build a visual analytics observatory that computes trends (both across topics and time points). Recent software engineering studies have analyzed various social media sources, such as blogs, forums, etc. [5, 3]. Recently, [4, 1] proposed the integration of social media into software development processes. However, few studies provide a practical solution to aggregate and analyze the wealth of software engineering information available in microblogs.

It is challenging to analyze the Twitter data. First, there are millions of users producing millions of tweets daily. As such, storing *all* tweets is technically impractical. Second, many Twitter users tweet about various subjects, not necessarily limited to software development, thus filtering out the irrelevant tweets is not trivial. Third, the massive number of tweets may contain useful *software-related trends* which are neither apparent nor readily available to Twitter users. Although one can use freely available tools to capture Twiiter feeds, their polling capacity are quite limited, making them unsuitable for constructing a large repository of software development-centric Twitter data.

To address the above challenges, there is a need for a scalable approach that is able to aggregate tweets made by Twitter users who are likely to tweet about software development. The approach needs to allow the users to perform *visual analytics* to better understand recent trends and developments from the data. Furthermore, it should allow for frequent periodic updates as new tweets and relationships 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 groups of topics. Longitudinal trends capture relative popularity of a topic at various points in time. We believe these trends could provide various insights to software developers, e.g., learning about the emerging topics that other developers care about, such as critical bugs, project management techniques, etc.

We took a user-centric approach of gathering the tweet data to be used by such an observatory. Specifically, 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

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 a few interesting trends that we capture from 70,000 Twitter feeds first logged since June 2011.

The structure of this paper is as follows. In Section 2, we present our proposed framework. In Section 3, we present our preliminary study and highlight some interesting topical and longitudinal trends. In Section 4, we present related studies. We conclude and describe future work in Section 5.

2. 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 tweet about software-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 the data.

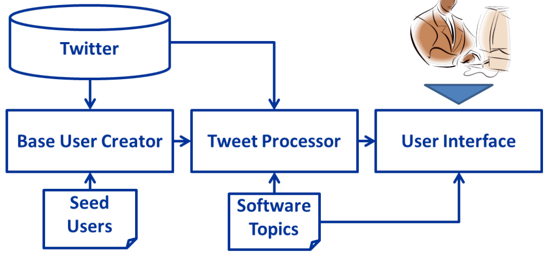


Figure 1: Proposed framework

2.1 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.

To accomplish this, our system takes in a set of seed users ($SEED$). In this case, the seed users are well-known Twitter users that actively tweet about software topics. These users are manually identified[1]; for example, *Jeff Atwood*, *Jason Fried*, and *John Resig*. We assume that any Twitter users who follow or are followed by 5 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 bidirectionally to substantially expand the user base. A set of seed users and their *friends* (the users whom they follow) and *followers* (the users who follow them) is denoted by $UBase$. Moreover, we continuously expand $UBase$ according to the changes in their friends and followers as they are periodically observed. We further elaborate on the update process in the next Section. As of May 9, 2012, there are approximately 70,000 unique users in $UBase$.

2.2 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. We automatically download all the latest tweets published by each user in $UBase$ 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[2], 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 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, stop-word removal, and stemming. We use whitespace and punctuation as the delimiters for tokenization, remove common English language stop words, and utilize Porter stemmer to reduce a word to its root form. We manually mark some technical synonyms and jargons that should not be stemmed

[1]http://www.noop.nl/2009/02/twitter-top-100-for-software- developers.html.
[2]https://dev.twitter.com/docs/rate-limiting

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[3]. 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 derive the 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.

2.3 User Interface

We implemented a simple web application using PHP and MySQL to provide access to the processed Twitter data. Our user interface supports both browsing and searching modalities. A snapshot of the user interface is shown in Figure 2. To facilitate browsing of software-related topics, the main user interface displays the three 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.

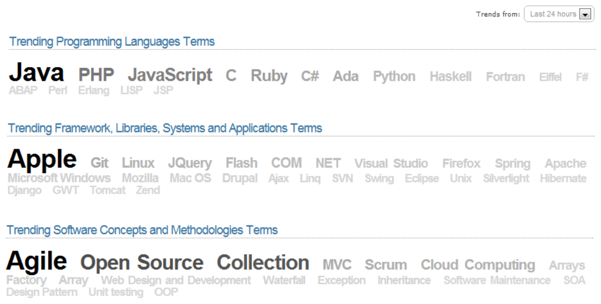
A user can click any topics in the three groups. After this, a line chart showing the number of tweets containing the topic over time would be shown. This chart represents the *longitudinal trend* of the topic. An example of this chart for "Scrum" is shown in Figure 3. Users could also investigate the context (e.g., the actual tweets or frequent words) at various points in time by clicking at the points along the line chart. An example of the resultant UI showing the words co-occurring frequently with "Scrum" on March 28, 2012 is displayed in Figure 4. Additionally, 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, separated by commas, can also be submitted together to generate multiple trend lines for comparison.

3. PRELIMINARY STUDY

3.1 Dataset

Our framework takes in a number of parameters: a set of s seed users and a set of t topics. For the set of seed users, we utilize the public list of popular Twitter users to identify the top-100 software developers that tweet. As described in Section 2, we include 100 software-related topics and divide them into 3 main categories. In this work, the values of s and t are 100. Next, we began polling Twitter

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


Trends from: Last 24 hours

Trending Programming Languages Terms

Java PHP JavaScript C Ruby C# Ada Python Haskell Fortran Eiffel F# ABAP Perl Erlang LISP JSP

Trending Framework, Libraries, Systems and Applications Terms

Apple Git Linux JQuery Flash COM NET Visual Studio Firefox Spring Apache Microsoft Windows Mozilla Mac OS Drupal Ajax Linq SVN Swing Eclipse Unix Silverlight Hibernate Django GWT Tomcat Zend

Trending Software Concepts and Methodologies Terms

Agile Open Source Collection MVC Scrum Cloud Computing Arrays Factory Array Web Design and Development Waterfall Exception Inheritance Software Maintenance SOA Design Pattern Unit testing OOP

Figure 2: User interface & topical trends

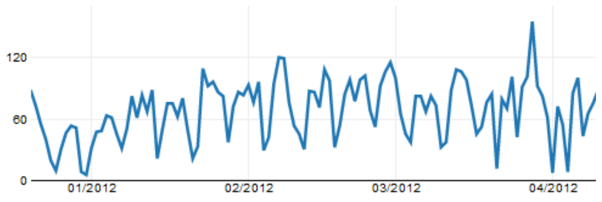
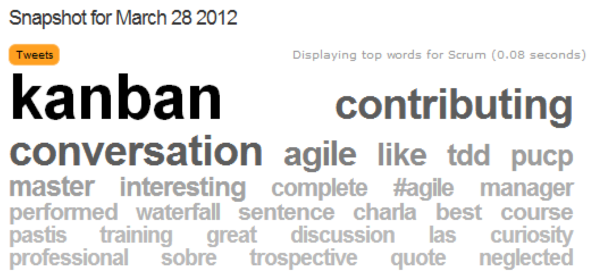


Figure 3: Longitudinal trend on "Scrum"


Snapshot for March 28 2012

Tweets

Displaying top words for Scrum (0.08 seconds)

kanban contributing conversation agile like tdd pucp master interesting complete #agile manager performed waterfall sentence charla best course pastis training great discussion las curiosity professional sobre trospective quote neglected

Figure 4: Background information about "Scrum" on March 28, 2012

data on a daily basis since June 2011. Given the limitation of the Twitter APIs, the oldest tweet we could retrieve thus far was published in July 2007. As of May 9, 2012, the whole dataset comprises approximately 70,000 unique users, 21 million follow links, and over 85 million tweets.

3.2 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 *Java*, *PHP*, and *Javascript* are the three popular languages mentioned by Twitter users during the last 24 hours of May 9, 2012. The corresponding topical trend interface is shown in Figure 2. Similarly, by comparing different software engineering concepts and methodologies, we find that *Agile*, *Open Source*, and *Collection* are the three most popular software-related concepts.

With longitudinal trend analysis, users can examine the popularity of a topic across time points. For example, from Figure 3, we can see that the tweeting frequency related to Scrum goes up and down periodically. A "sawtooth" pattern reflects a variation of activities during a typical working week. This pattern can be observed in the trend lines of other generic topics, e.g., Javascript, PHP, etc. Next, we can also identify events corresponding to the unusual peaks

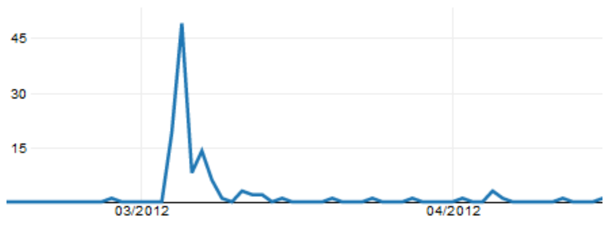


Figure 5: Longitudinal trend about "GitHub security"

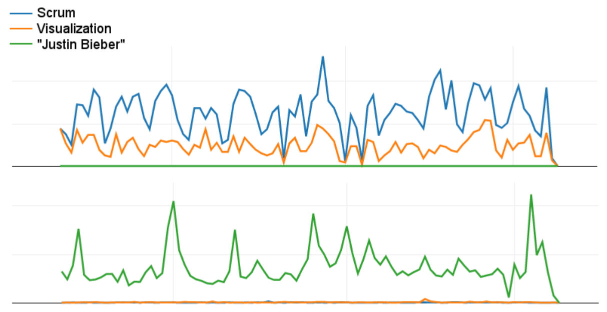


Figure 6: Comparing the longitudinal trends in the developer-centric (top) and general repositories (bottom)

or dips in the trend line. For example, we notice that there is an unusual peak in the number of tweets on March 28, 2012. Upon further inspection, we find that there was a rigorous discussion in social media about the differences between two agile development methods, Scrum and Kanban. When a more specific topic is used, a more "bursty" trend line may be observed. For instance, Figure 5 shows a longitudinal trend "GitHub security" related to the security issue of GitHub, a popular online project hosting site. In this case, we can clearly see a sudden peak on March 5, 2012 corresponding to the hacking incidence.

Figure 6 displays a comparison between three trend lines about "Scrum","visualization", and "Justin Bieber", observed in the proposed software-developer centric repository (top), compared to those in a general (bottom) repository[4], built using a different seeding set who are those interested in general topics. As we can see, our framework effectively filters out noisy topics which are less relevant to the software development community, e.g., Justin Bieber, etc., while reasonably captures the relevant trends.

4. 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 [4, 1]. Pagano and Maleej analyzed how open source communities blog [5]. Gottipati et al. built a semantic search engine to effectively find answers in software forums [3]. 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.

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 [2]. Surian et al. and Hong et al. analyzed developer socio-technical network in SourceForge.Net [6].

Twitter Analytics Our solution can be contrasted with Twitter Analytics applications, such as the Archivist[5]. Unlike the Archivist, our tweet observatory specifically focuses on the software engineering domain. Moreover, our interface encourages more serendipitous discovery of other interesting software engineering topics through a topical trend browsing. Since we exclusively focus on the software development community, our data are also less prone to noise than the Archivist's.

[4]http://research.larc.smu.edu.sg/palanteer/

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

5. CONCLUSION AND FUTURE WORK

In this work, we propose a web application that collects, processes, and presents the Twitter data generated by a selected and evolving set of software developers to the end users as trends via the topical and longitudinal trend analysis interface. We have also found some interesting trends in our preliminary experiments. In the future, we plan to track the Twitter feeds in realtime and perform more sophisticated analysis on the data. 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 massive microblog data.

6. ACKNOWLEDGMENTS

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

7. REFERENCES

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

[2] C. Bird, A. Gourley, P. T. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *MSR*, pages 137–143, 2006.

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

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

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

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

[7] Y. Tian, P. Achananuparp, I. Lubis, D. Lo, and E.-P. Lim. What does software engineering community microblog about? In *MSR*, 2012.