2012 28th IEEE International Conference on Software Maintenance (ICSM)

Automatic Classification of Software Related Microblogs

Philips K. Prasetyo, David Lo, Palakorn Achananuparp, Yuan Tian and Ee-Peng Lim
School of Information Systems
Singapore Management University, Singapore
{*pprasetyo,davidlo,palakorna,yuan.tian.2012,eplim*}*@smu.edu.sg*

***Abstract*—Millions of people, including those in the software engineering communities have turned to microblogging services, such as Twitter, as a means to quickly disseminate information. A number of past studies by Treude et al., Storey, and Yuan et al. have shown that a wealth of interesting information is stored in these microblogs. However, microblogs also contain a large amount of noisy content that are less relevant to software developers in *engineering* software systems. In this work, we perform a preliminary study to investigate the feasibility of automatic classification of microblogs into two categories: relevant and irrelevant to engineering software systems. We extract features from the textual content of the microblogs and the titles of any URLs mentioned in the microblogs. These features are then used to learn a discriminative model used in classifying relevant and irrelevant microblogs. We show that our trained model can achieve a promising classification performance.**

I. Introduction

Microblogging has recently become a popular means for people to exchange opinions and disseminate information. Microblogs often highlight recent or even real time contents that are of interest to many. One of the most popular microblogging services to date is Twitter. Due to its vast user base and popularity, we focus on microblog messages published by Twitter users in this work.

In Twitter, a microblog message, also known as *tweet*, may contain up to 140 characters. In addition, a tweet can be embedded with URLs and media, such as images, videos, etc. Users can also subscribe to or *follow* other users' message feeds, forming a network of users. Whenever a Twitter user publishes a tweet, it will be automatically transmitted to their *followers*' subscribed feeds. Thanks to the realtime and informal nature of Twitter, more than 340 million tweets are generated by over 140 million active users every day[1].

People in the software engineering community also publish tweets regularly. Guzzi et al., Begel et al., and Treude and Storey propose ways to integrate social media and microblogging with software development and IDEs [6], [3], [18]. Bougie et al. and Tian et al. investigate what kinds of microblogs are generated by users [4], [16]. By subscribing to others' message feeds, developers could learn about various kinds of information useful to their software engineering activities, for example, new tools, sample code snippets, tips, etc.

Unfortunately, the majority of tweets are neither informative [11] nor related to software engineering topics. Moreover, many software-related microblogs are neither relevant nor helpful to *engineering* software systems. Tian et al. [16] manually classified software-related microblogs and found that many of them are about job advertisement. Thus, interesting microblogs that are relevant to engineering software systems are often buried among a plethora of irrelevant ones.

In this work, we propose a preliminary framework to identify *relevant* microblogs (those that could be helpful in engineering software systems) from *irrelevant* ones. Our framework consists of three components: Webpage crawler, text processor, and machine-learning classifier. For any tweets with embedded URLs, we crawl the content of the URLs mentioned in them. Next, we process the contents of the tweets and the embedded URLs to construct the feature sets used to train the classifier. Then, we train the classifier using the feature sets to build a discriminative model. The model is then used to classify the relevancy of unlabeled tweets.

We perform a preliminary evaluation on a set of 300 software-related tweets previously studied by Tian et al. [16]. These tweets contain hashtags (i.e., user-defined tags that are embedded in tweet content, mostly for specifying the tweet's topics) related to software development. Still, only 47% of the tweets are relevant to engineering software systems. Our trained model, on the other hand, achieves a promising performance for the same set of tweets.

The contributions of this work are as follows:

1) We propose a new problem of automatic categorization of tweets as relevant or irrelevant to engineering software systems.
2) We propose a text processing framework that extracts textual features from both tweets and embedded webpages. These features are then used to train a discriminative model that can be used to classify relevant and irrelevant tweets.
3) We have conducted a preliminary experiment on a set of 300 tweets. Our tweet classifier performs reasonably well, achieving 74.67% accuracy, 76% precision,

[1] http://thenextweb.com/socialmedia/2012/03/21/twitter-has-over-140-million-active-users-sending-over-340-million-tweets-a-day

978-1-4673-2312-3/12/$31.00

67.38% recall, and 71.43% F-measure.

The structure of this paper is as follows. First, we elaborate our proposed framework in Section II. Then, we present the experiment results in Section III and discuss the related work in Section IV. Lastly, we conclude the paper and mention future work in Section V.

II. Proposed Approach

The proposed framework, shown in Figure 1, consists of three major components: Webpage crawler, text processor, and tweet classifier. In the training stage, we process a set of manually-labeled tweets as either relevant or irrelevant and use them to train a classifier. A discriminative model is an output of the training phase. Then, we use the discriminative model to classify unlabeled tweets at the testing phase.

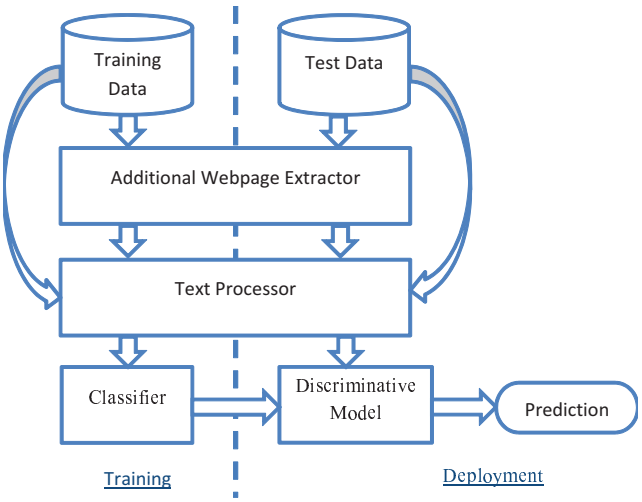

Figure 1. Proposed Framework

A. Webpage Crawler

As a tweet can only contains at most 140 characters, Twitter users often use URL shortener services, such as *bit.ly*, to encode the original URLs before embedding them in the tweet content. The content of these URLs could be helpful in classifying the relevancy of the tweet itself. Therefore, we create a crawler to detect a shortened URLs (using a simple regular expression) in the tweet content, expand it by checking HTTP headers so as to derive the original URL, and collect its textual content.

B. Text Processor

The text processor component is used to pre-process the contents of tweets and embedded webpages so as to extract a set of textual features to be used in the classification. For each tweet, we remove common stopwords from its content using NLTK[2] stopword list. Next, we stem (reduce a word to its root form, for example, both "reading" and "reads" share a common root form "read") each word in the tweet using the Porter stemmer algorithm [13]. For each embedded webpage, we apply the same pre-processing steps to its title. Finally, we extract single-word tokens from the pre-processed tweet and webpage texts and combine them as a feature set.

[2]http://nltk.org/

C. Classifier and Discriminative Model

We represent each labeled (and unlabeled) tweet as a feature vector where each feature is a single-word token having its frequency in the tweet as its weight. Given a set of feature vectors of the training data, various classifiers can used to train a discriminative model. In this study, we use Support Vector Machine (SVM) as it has been commonly used in other text mining studies in software engineering, e.g., [9], [15]. The trained discriminative model is then used in the deployment phase to classify unlabeled tweets.

III. Preliminary Experiments

In this section, we describe our dataset, research questions, preliminary results, and threats to validity.

A. Dataset

Tian et al. [16] collected 300 tweets containing either one of the following 9 hashtags: #csharp, #java, #javascript, #.net, #jquery, #azure, #scrum, #testing, and #opensource. They then manually classified the tweets into 10 categories: commercials, news, tools & code, Q&A, events, personal, opinions, tips, jobs, and miscellaneous. In this study, we use Tian et al.'s dataset in our experiment. Since the goal is different in our study, we manually re-label the 300 tweets as either relevant or irrelevant. If a tweet is potentially interesting to a developer for his/her task of engineering a software system, we label it as relevant, otherwise we label it as irrelevant. We find that 141 (47%) of the microblogs are relevant while 159 (53%) of them are irrelevant.

B. Research Questions

We would like to answer the following research questions:

- RQ1 How effective is the proposed framework in classifying the tweets as relevant or irrelevant?
- RQ2 Given the categories of software engineering tweets in Tian et al., how many tweets in each category is relevant and how many are irrelevant? How many tweets of each category are correctly classified?
- RQ3 What is the sensitivity of the proposed framework when less training data is available?

Table I
OVERALL EFFECTIVENESS

Accuracy	74.67%
Precision	76%
Recall	67.38%
F1-Measure	71.43%

C. RQ1: Overall Effectiveness

To measure the accuracy of the proposed framework, we perform a 10-fold cross validation. We divide 300 tweets into 10 folds containing 30 tweets each. We use 9 of them for training and 1 for testing. We repeat the process 10 times considering different group as the test set.

We measure the effectiveness of the proposed framework using the standard measures in text classification: accuracy, precision, recall, and F-measure. Accuracy measures the percentage of correctly classified tweets. It is defined as

$$\frac{tp + tn}{tp + tn + fp + fn}$$

where tp, tn, fp, and fn are true positive, true negative, false positive, and false negative, respectively. Precision is the fraction of true relevant tweets among predicted relevant tweets, is defined as

$$\frac{tp}{tp + fp}$$

Recall measures the percentage of relevant tweets captured by the model. It is defined as

$$\frac{tp}{tp + fn}$$

F-measure is the harmonic mean of precision and recall and is defined as

$$\frac{2 \times precision \times recall}{precision + recall}$$

Table I shows the overall effectiveness of our proposed framework. The results look promising. Our discriminative model achieves 74.67% accuracy, 76% precision, 67.38% recall, and 71.43% F-Measure given that only the contents of tweets and webpages' title are used to train the classifier.

D. RQ2: Effectiveness per Tweet Category

We summarize the distribution of tweets across Tian et al.'s categories [16] in Table II. As we can see, over 85% of tweets in tools & code, tips, and Q&A are relevant to engineering software systems while no tweet in personal, jobs, and miscellaneous is relevant.

We also investigate the effectiveness of our framework on each category of tweets. Table III shows the per-category performance of the proposed framework.

The results show that our framework achieves high F-measure in the two highly relevant category, i.e., tools & code, and Q&A. On the other hand, it does not perform as well in the other highly relevant category, i.e., tips. Although the model achieves 100% recall when classifying tips-related tweets, the overall performance suffers from low precision. Furthermore, our framework also performs very well in classifying tweets in highly irrelevant categories, i.e. personal, jobs, and miscellaneous. The most difficult category to classify is events because many tweets in this category are ambiguous. We expect that the classification performance in this category can be further improved by incorporating more textual features from the URL contents.

Table II
RELEVANCE PER CATEGORY OF TWEETS

Category	**% Relevant**
Commercials	40%
News	29.5%
Tools & Code	100%
Q&A	86.4%
Events	45.5%
Personal	0%
Opinions	42.9%
Tips	100%
Jobs	0%
Misc.	0%

Table III
ACCURACY PER CATEGORY OF TWEET CLASSIFICATION

Category	**Accuracy**	**Precision**	**Recall**	**F-Measure**
Commercials	60%	50%	50%	50%
News	54.5%	61.5%	34.8%	44.4%
Tools & Code	79.5%	79.5%	100%	88.6%
Q&A	79.6%	84.2%	91.4%	87.7%
Events	45.5%	20%	33.3%	25%
Personal	93.8%	0%	0%	0%
Opinions	76.2%	55.6%	83.3%	66.7%
Tips	48.5%	48.5%	100%	65.3%
Jobs	100%	0%	0%	0%
Misc.	72%	0%	0%	0%

E. RQ3: Sensitivity Analysis

We also investigate the sensitivity of our framework on the number of training data. Since we have a small dataset, we perform k-fold cross validation using different k values from 2 to 9. Smaller k values use fewer data points for training. The performances of k-fold cross validations for different values of k, shown in Table IV, suggests that our framework is robust on different numbers of training data.

F. Threats to Validity

Threats to internal validity relates to experimenter bias. In our study, to create the ground truth, we manually assign the

Table IV
RESULTS OF K-FOLD CROSS VALIDATIONS FOR DIFFERENT K

k	**Accuracy**	**Precision**	**Recall**	**F-Measure**
9	75.43%	75.19%	70.92%	72.99%
8	74.29%	74.62%	68.79%	71.59%
7	73.98%	74.05%	68.79%	71.32%
6	74.33%	73.88%	70.21%	72%
5	73.67%	74.22%	67.38%	70.63%
4	75%	75.78%	68.79%	72.12%
3	74.67%	74.44%	70.21%	72.26%
2	75%	75.78%	68.79%	72.11%

tweets to two categories: relevant (to engineering software systems) and irrelevant. As with any other manual labeling tasks, human judgment can be biased or inconsistent.

Threats to external validity relates to the generalizability of our findings. In this study, we only evaluate our framework on 300 microblogs in Twitter. In the future, we plan to reduce this threat of external validity by conducting an experiment using a larger dataset.

Threats to construct validity refers to the suitability of our evaluation measures. We use the standard 10-fold cross validation and evaluate the effectiveness of our framework using the standard evaluation measures used in text classification, namely, accuracy, precision, recall, and F-measure. Thus, we believe there is little threat to construct validity.

IV. Related Work

Social Media and Software Engineering. Recently, a number of studies highlight an interesting direction of research on how social media could help software development. Storey highlights the roles that various social media, ranging from question and answer sites, microblogging sites, social coding sites, etc., could play in improving software development [14]. A number of studies by Guzzi et al., Begel et la., and Treude et al. have proposed various ways of integrating social media to software development and integrated development environments [6], [3], [18]. Pagano and Maalej investigate how developers blog [12]. Treude et al. investigate and manually categorize a few hundreds questions in StackOverflow [17]. Gottipati et al. build a search engine over software question and answer forums by inferring semantic tags [5]. Bougie et al. and Tian et al. investigate a few hundred microblogs and manually categorize them [4], [16]. Achananuparp et al. build an observatory of software microblog trends [1]. In this study, we extend the prior work by Bougie et al. and Tian et al. by proposing a framework to automatically categorize software microblogs into two classes: relevant and irrelevant to engineering software systems.

Text Classification for Software Engineering. A number of existing studies employ text classification for various software engineering tasks. Antoniol et al. use text classification to predict if a change requests is a bug report or a feature request [2]. A number of studies use text classification to predict the severity of bug reports. These include the study by Menzies and Marcus [10] and Lamkanfi et al. [8], [9]. Other studies use text classification to identify duplicate bug reports, e.g., [15]. In this work, we use text classification to automatically identify relevant and irrelevant microblogs. We extract the content of both the microblogs and URLs embedded in the microblogs (we only use the URL's title in this preliminary study) to form textual features that are used to train a microblog classifier.

V. Conclusion and Future Work

In this work, we propose a text mining framework to automatically categorize microblogs into those that are either relevant or irrelevant to engineering software systems. We extract a set of features from the content of the tweets as well as the ULRs embedded in them. Then, we train a discriminative model using support vector machine (SVM) as the classifier. The discriminative model is used to classify unlabeled microblogs as either relevant or irrelevant. The preliminary evaluation on a set of 300 tweets suggests that the proposed framework is promising.

In the future, we plan to extract more features from URL contents and conduct a larger scale evaluation using a finer grained categories, e.g., the 10 categories proposed by Tian et al. [16]). Moreover, we would like to investigate other classifiers that produce human readable models, e.g., decision tree [7], or a spam-filter-like Bayesian approach.

Acknowledgement

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

References

[1] P. Achananuparp, I. Lubis, Y. Tian, D. Lo, and E.-P. Lim, "Observatory of trends in software related microblogs," in *ASE (Tool Paper), also available at: http://research.larc.smu.edu.sg/papers/observatory.pdf*, 2012.

[2] G. Antoniol, K. Ayari, M. D. Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement?: a text-based approach to classify change requests," in *CASCON*, 2008.

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

[4] G. Bougie, J. Starke, M.-A. Storey, and D. German, "Towards understanding Twitter use in software engineering: Preliminary findings, ongoing challenges, and future questions," in *International Workshop on Web 2.0 for Software Engineering*, 2011.

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

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

[7] J. Han and M. Kamber, *Data Mining Concepts and Techniques*, 2nd ed. Morgan Kaufmann, 2006.

[8] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the severity of a reported bug," in *MSR*, 2010.

[9] A. Lamkanfi, S. Demeyer, Q. Soetens, and T. Verdonck, "Comparing mining algorithms for predicting the severity of a reported bug," in *CSMR*, 2011.

[10] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," in *ICSM*, 2008.

[11] M. Naaman, J. Boase, and C.-H. Lai, "Is it really about me?: message content in social awareness streams," in *Proceedings of CSCW '10*, 2010.

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

[13] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[14] M.-A. Storey, "The evolution of the social programmer: Social media and software engineering (keynote speech)," in *MSR*, 2012.

[15] C. Sun, D. Lo, X. Wang, J. Jiang, and S.-C. Khoo, "A discriminative model approach for accurate duplicate bug report retrieval," in *ICSE (1)*, 2010, pp. 45–54.

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

[17] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?" in *ICSE*, 2011.

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