MEME – Toward a Method for EMotions Extraction from GitHub

Karl Werder
paluno, University of Duisburg-Essen, Germany
karl.werder@paluno.uni-due.de

Sjaak Brinkkemper
Utrecht University, The Netherlands
s.brinkkemper@uu.nl

ABSTRACT

Software engineering researchers are increasingly interested in the role of emotion during software development. While general tools are available to extract emotions from textual data, these perform poorly in the domain of software engineering. Hence, this paper develops MEME – a Method for EMotion Extraction. Using GHtorrent and GitHub as data sources, the paper presents an implementation of the method. The evaluation results suggest a better performance of MEME in contrast to Syuzhet R package emotion analysis.

1 INTRODUCTION

Software engineering researchers developed an increasing interest in the role of emotions [4, 14], as these influence developer's and project's performance [4, 24]. While prior studies often capture emotions through surveys and interviews, more recently scholars explore the use of large data sets as a source for emotion extraction. Data sets in software development often analyze comments on work progress, e.g. commits or issues [6, 9, 19], discussion forums and mailing lists [20], or comments in the source code versioning tools [14]. In contrast to surveys and interviews that often rely on synthetic generation of data through scholars' engagement, the aforementioned examples rely on data generated by developers independently of any research study. Hence, a more accurate view of developers' true emotions at the time can be analyzed.

However, automatic extraction of emotions is a challenge. While tools are available in order to process natural language, these are often established and tested for newspaper articles. Given the fact that natural language is domain specific, these general tools vary in their performance and accuracy when applied to individual domains, such as software development [8]. Hence, further research is needed toward tailored methods and tools that assess emotion extraction in the domain of software development. We formulate the following research question: *How to design a method for emotion extraction from software repositories?* Therefore, we develop MEME – a Method for EMotion Extraction based on an example GitHub dataset. In order to evaluate MEME, we implement the proposed algorithm. Results are compared to manual coding and the results of a popular emotion extraction toolkit.

SEmotion'18, June 2, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5751-7/18/06…$15.00
https://doi.org/10.1145/3194932.3194941

2 RELATED WORK

Prior research investigating emotions in software development apply varying research methods. Studies apply questionnaires [4], interviews [5], case studies [9], and repository mining approaches [6, 14, 19, 23].

The results of an in-depth study with two developers suggest that the influence of emotions on programming performance is mediated by the developer's focus [5]. Happy developers are better problem solvers has been suggested by a study with 42 software engineering student participants [4]. A case study with nine open source software (OSS) projects investigates the sentiment of developers' issues and tickets [9]. Results find underlying sentiments within projects and find emotional analysis to be more applicable to text from issues rather than tickets.

Prior studies applying a repository mining approach investigate issue reports from Jira by the Apache software foundation. Their results suggest that human raters identify emotions from such documents [14]. Yet, the accessibility and agreement varies with each emotion. A lexical sentiment analysis study on commit comments from top starred GitHub projects by programming language finds that comments in java carry more negative sentiment in contrast to other languages. Further, the study suggests that comments written on Mondays are more negative [6]. In contrast, a study investigating commit logs for GitHub projects finds that Tuesday's comments have the most negative sentiment [19]. The study also finds a strong correlation between the expressed sentiment and the number of files changed. A longitudinal analysis of positive emotional displays in (OSS) development teams shows their decay over time [23].

3 NATURAL LANGUAGE PROCESSING

Multiple tools are available in order to conduct an emotion analysis for textual data. We identify two central solutions that are often used by scholars. While other tools are available (e.g. SentiStrength, NLTK, Alchemy [8]), we focus on tools from the R community as these have been largely adopted by the scientific community [17] and are easily accessible and reusable as an open source tool. First, Sentiment R Package developed by Jurka [10]. The package uses the Wordnet-Affect sentiment lexicon [9]. The

lexicon is freely distributed under Creative Commons License and it contains 1512 unique words in the corpus. Second, Syuzhet R Package developed by Jockers [7]. The package uses the NRC Lexicon with 14182 unique words [13].

The identified two techniques provide different classifications of emotions. The Sentiment R Package identifies six classes of emotions using a discrete emotional framework [1]: anger, disgust, joy, surprise, fear, and sadness. The. The Syuzhet R Package identifies eight classes of emotions as suggested by Plutchik's wheel of emotions [16]: joy and sadness, trust and disgust, fear and anger, surprise and anticipation. This dimensional framework of emotions is balanced with 4 positive and 4 negative emotions, as they group emotions into emotional pairs, such as joy and sadness.

4 A METHOD FOR EMOTION EXTRACTION

MEME is a Method for the extraction of emotions from software developers' comments in software repositories. In this section, the methods main activities are described along the data collection, data sampling and cleaning, and emotion extraction.

4.1 Data Collection

In order to identify a consistent and representative dataset, we rely on GitHub as one of the largest OSS repository available. We find three main approaches to extract data from GitHub: i) using the APIs provided by GitHub [18], ii) using GHTorrent, a mirror of all GitHub projects [3], or iii) using GitHub Archive, an archived copy of GitHub timeline[2].

Below, we explain challenges associated with these approaches and justify our selection. The GitHub API is characterized by fast response, ease-of-use, availability of multiple libraries to use the API, and consistent data. However, it has a limit of 5000 requests per hour [18], making it harder to retrieve a large dataset of GitHub projects. On the contrary, GHTorrent has no limit and is available in MongoDB structure. However, it has two challenges: i) it is very slow due to the dataset being hosted on a slow server at the time of collecting the dataset; and ii) it misses linkage information to connect different attributes. Lastly, GitHub Archive stores the data in JSON format and splits the data set per hours, requiring the implementation of a JSON parsers before further data processing.

Hence, we opt for a hybrid approach to collect the data. GHTorrent provides a great way to retrieve all the events associated with repositories ranging from issues, pull request, followers and comment events. In a similar vein, GitHub API is able to retrieve additional information, such as repositories' commits and users, which were extremely slow to retrieve from the GHTorrent dataset.

As the study aims to extract emotions from textual data, repositories with users' comments on pull requests, commits, or issues were collected; GHTorrent was analyzed based on the developers' comments on the retrieved repositories. 440552 projects had comments on their commits, while 149162 projects had comments on their pull requests. In sum, 538115 projects contained comments ranging from 1 comment up to ten thousand comments, suggesting the need for further data sampling.

4.2 Data Sampling

In order to extract emotions, we need unique and active projects. Hence, we clean the data, remove duplicates, and select active projects. In addition, we ensure representativeness of the dataset by defining clear inclusion and exclusion criteria. The following criteria apply in order to remove personal, inactive, or projects used as storage and filter for active projects [11, 20–22]:

Remove duplicates: We identified and removed duplicated entries for repositories, commits, and comments from GHTorrent.

Number of comments: We selected projects with at least 50 comments either on commits or pull requests. This textual data allows extracting the underlying developer's emotions using emotion analysis techniques [20].

Missing commits: We excluded projects with unknown number of commits, as this indicated deleted projects. While GHTorrent retrieved the information before a project got deleted, we are not able to extract additional information through the GitHub API.

Missing programming language: We excluded projects that miss a programming language. While this is not a mandatory parameter in GitHub, its absence indicates storage projects.

Missing committers: We excluded projects with much missing committer's information. As these are central information in order to link an emotion to a developer, projects that have more than 25% missing committers were omitted from the selection.

Missing lines of code (LOC): Each commit includes the lines of code added, edited or removed. We excluded those projects, where more than 25% of LOC information were missing.

Active Projects: We selected projects with at least 6 developers, lasting for at least 6 months, having more than 24 commits, and at least two commits per month [22].

Our initial set of 538115 projects contained textual communication in the form of developers' comments. Following the removal of duplicates and projects with less than 50 comments, we find 6622 projects in this step. Thereafter, we removed projects with missing commits, having a remainder of 6467 projects. Filtering projects based on number of developers, duration, number of commits, missing programming language, missing committers' information and missing LOC led to a remaining 2746 projects. Finally, when filtering for *Active Projects*, we find 1134 projects.

4.3 Emotion Analysis Techniques

Following the identification of suitable projects, we describe the extraction of emotions in further depth. We implemented a tailored algorithm in order to improve the extraction of emotions from GitHub comments. The Syuzhet R package is popular and suggests good results, and therefore, forms our point of reference. Prior work suggested small adaptations to it [9]. Scholars removed punctuation and replaced them with empty strings instead of spaces [9]. While the use of spaces might lead to the extraction of additional words, these may not accurately reflect the sentiment of the developers and thus, negatively impact the emotion extraction.

2

However, two challenges remain when developing a domain specific emotion extraction. First, the identified algorithms both [7, 10], neglect negating terms during emotion extraction. For instance, the following two example sentences would result in the same extracted emotions, despite a negating term in one of them: 1. "I really hate working on this project"; 2. "I don't really hate working on this project". Second, developers' comments on GitHub often contain exceptions or error messages that could negatively impact emotion extraction. Prior scholars noted this challenge, highlighting the fact that developers tend to paste compiler error messages, warnings, or exception messages in their comments [9]. Likewise, others note the urgency to eliminate domain-specific keywords which could affect the reliability of the results [15]. For example, the sentence "I got the following error: Bad file number." would be connotated with a negative emotion and hence, would not reflect the developer's emotion correctly.

Error Messages. When analyzing the error messages, we observed that most of the sentiments associated with lines of code and compiler messages were nouns. Following prior recommendations [15], we removed all nouns from our emotion extraction in order to improve results. In addition, we developed a list of domain specific keywords. Keyword lists have been successfully used by other studies in sentiment mining [9]. This list was developed by analyzing the comments and the extraction of emotions. The following keywords negatively influenced the extracted emotion by the algorithm: *attach, attachment, bash, build, broke, merge, error, case, catch, compact, console, content, inconvenient, render, bug, static, generic, variable, main, enhanced, default, production, remove, supported, unsupported, execution, gateway, system, error, argument, action*.

Negation. Negations were found to be critical for the analysis. Hence, a negation detection mechanism was introduced and a negation window was added. Thus, once a negation is indicated, all the words' emotions within this window will be negated. In the course of this study, the negation window was chosen to be 4 words. Hence, any emotion that is 4 steps before or after the negating word would be inverted. While sentiment negation is trivial, it requires a dimensional emotional framework.

Integration. Finally, we integrate the proposed modifications with common NLP steps into the algorithm. Thus, we start by reading the comments, split the sentences, tokenize each sentence, and apply Part-Of-Speech (POS) tagging. Stanford CoreNLP, an integrated framework providing a set of linguistic analysis tools was used to conduct sentence splitting, tokenization, and POS tagging [12]. Given the prior POS tagging, we select adjectives, adverbs, and verbs. We extract the emotions from the selected words using the corresponding emotions as documented in the NRC lexicon. Thereafter, we detect negations and identify the opposite emotions for each word within the negation window. Last, we sum all emotions and standardize the scores per sentence by dividing the sum over the number of sentences within each comment. The implemented algorithm is shown in Algorithm-1.

Algorithm-1: Tailored Emotion Extraction algorithm

```
1:  negation_window ← 4
2:  for each comment c ∈ comments do
3:      c ← tolower(c)
4:      c ← tokenizer(c)
5:      sentences ← sentence_splitter(c))
6:      emotion ← ϕ
7:      for each sentence s ∈ sentences do
8:          part_of_speech ← pos(s)
9:          negations_positions ← negations(s)
10:         words ← split(s)
11:         for each word w ∈ words do
12:           if (part_of_speech(w)== 'Adjective' OR
13:           part_of_speech(w)== 'Adverb' OR
14:           part_of_speech(w)== 'Verb') then
15:             if w ∉
16:             within_negation_win(negation_window,ne
17:             gation_positions) then
18:                 emotion ← emotion +
19:                 Get_NRC_Emotion(w)
20:             end if
21:           end if
22:         end for
23:     end for
24:     emotion ← emotion/size_of(sentences)
25: end for
26: return emotion
```

5 MEME TOOL

We developed a tool that implements MEME and collects data associated to GitHub projects, pre-processes the data, stores the data, and applies emotion analysis. In this section, we present the tools architecture and data collection cycle.

5.1 Architecture

The tool has been designed using a component-based architecture that integrates R for data analysis and MySQL for data storage (Figure 1). The tool is developed in Java with EmotionAnalyzer block, two components that access external data sources (GHTorrentMapper and GitHubAPIMapper), and a component for persistent data storage connecting the MySQL database.

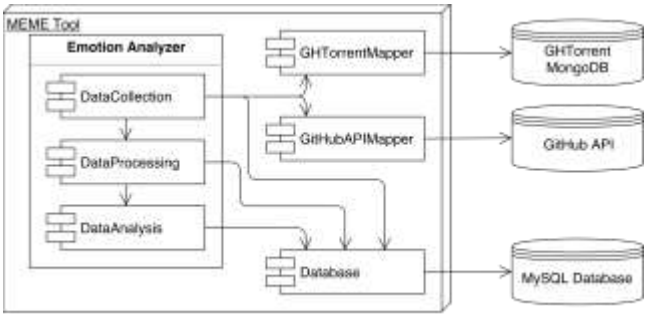

Figure 1: MEME tool architecture.

First, the *DataCollection* component is a Java Library responsible for collecting the dataset of the projects and storing them into the MySQL Database. The component transforms the data into a structured format and stores it in a relational database instead of unstructured documents format MongoDB. Second, the *DataProcessing* component integrates an R package that provides emotion analysis techniques and stores the outcomes into the database. The Java component implements the tailored designed emotion analysis technique in order to provide a more domain specific solution. Third, the *DataAnalysis* component creates descriptive statistics and visualizes the results. The analysis component was developed based on R and MySQL.

The *GHTorrentMapper* component accesses the MongoDB from the GHTorrent Project and extracts relevant data. Such data include the followers, pull requests & comments, issues & comments, commit comments. The *GitHubAPIMapper* component implements the GitHub API in order to complement GHTorrent data with current information on the users and project commits. The *Database* component implements a persistent data storage from all components of the *EmotionAnalyzer* block.

5.1 Data Collection Cycle

The tool's development is divided into four activities: i) crawling for repository, commit, and user data; ii) persistent storage of such data; iii) crawling for repository activity; iv) persistent storage of repository activity data. While the repository, commit, and user data are collected through the GitHub API, most of the data are extracted from the MongoDB. The Mongo DB supplies commit comment, issues and their comments, pull requests and their comments, as well as information on the followers. Figure 2 provides an overview of the data collection cycle.

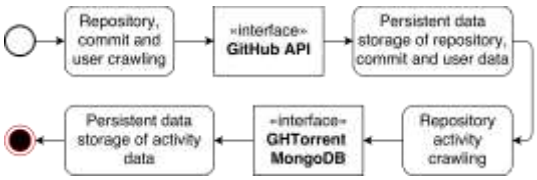

Figure 2: MEME data collection cycle.

6 EVALUTATION

Following the method development and tool implementation, we evaluate our proposed algorithm. We randomly select 60 comments from our dataset[2]. We noticed that negation exists in a 29 out of the 60 comments. That represents 48% of the comments. Therefore, it is important to take negation into consideration, when performing emotion analysis using GitHub data. Codes and exception messages were presented in 17% of the sampled comments. Also, a group of domain specific words exist that misled the algorithm.

In order to account for domain specific words in general, and error messages in particular, we collected 1163 compiler's error and exception messages from the dataset of 1134 projects. While most error messages concerned a specific programming languages (i.e. c++, java, http, js, php, python, and ruby), we found 129 messages that are common and apply to multiple languages. Accounting for these exception messages, we re-ran the Syuzhet algorithm to check the outcomes. However, the results suggest that this solution fails to detect most of the exceptions within the randomly collected comments. Hence, we opted for an alternative approach using keywords (cf. Section 4.3).

We test our proposed algorithm using the sampled comes and manually review the results for plausibility and correctness. We compare the tailored algorithm with the outcomes from Syuzhet R Package to evaluate performance differences. A comparison on the general sentiment level suggests that both algorithms agree on 60% of the cases. When reviewing on the emotion level, the manual review suggests that the tailored algorithm was better in 58% of all cases in contrast to 28% from the Syuzhet R Package. 14% of the cases were on par by both algorithms. We provide an example of the different emotions extracted by Syuzhet and our approach. Consider the following comment: "*Well, we don't really have the resources to maintain a powerpc option. But I **encourage** you do make your branch *the* PPC branch of Homebrew.We'll even link you from the **relevant** places. Sound **good**?*" Terms in bold express emotions extracted by the tailored approach; underscored terms are examples for nouns that had been beneficially excluded by the tailored approach. On the one side, Syuzhet counts 3 occasions of joy and 4 of trust, on the other side, the tailored approach extracts only 2 occasions of joy and 3 occasions of trust. While Syuzhet extracts more emotions, the tailored algorithm is more accurate. Since the frequency of emotions can be an indicator for their strength, the tailored approach is preferable.

We also investigate the extracted emotions for a sample project in order to compare results with other studies. We investigate the project Microsoft TypeScript. TypeScript is a programming language for application-scale JavaScript. The project has 44413 comments by the time of data collection. When comparing the emotions extract from Syuzhet with our tailored algorithm, we find that Syuzhet extracts 30% of neutral comments and our tailored algorithm extract 50% neutral comments. Therefore, the results from the tailored algorithm are in line with prior studies [9]. A visual comparison of the extracted emotions over time shows similar patterns for both algorithms.

Hence, the evaluation through manual reviews and a comparison of descriptive statistics on a project level provide a similar picture. While the tailored approach extract less emotions, the extracted emotions provide similar patterns over time and more accurate reflections of the emotions when assessing individual results. In addition, we noticed the frequent use of negations within developers comments that may results in severe misinterpretation of results if not adequately addressed.

[2] The sample of 60 comments (confidence level of 56%) is accessible via https://goo.gl/ywYXTj.

7 CONCLUSION AND FUTURE RESEARCH

We developed MEME, a Method for EMotion Extraction. We use GitHub as an example software repository, given its popularity and prior research results. In order to demonstrate and automate the emotion extraction method, we developed a tool. The tool assists with the emotion extraction from GitHub and builds on existing research infrastructure. A limitation for the adoption of the tool is its boundary conditions. We built the tool using functions from Syuzhet R package and the NRC Lexicon. As future work, we intend to use the tool to collect and process a large dataset. The tool collects various data points associated with GitHub's projects. Such data can be used to investigate and understand software development projects in greater depth and develop variance-theoretical models for emotional contagion.

REFERENCES

[1] Ekman, P. and Davidson, R.J. 1994. *The nature of emotion: Fundamental questions. Series in affective science.* Oxford University Press.

[2] GitHub Archive: 2017. *https://www.githubarchive.org/*. Accessed: 2017-11-30.

[3] Gousios, G. 2013. The GHTorent dataset and tool suite. *2013 10th Working Conference on Mining Software Repositories (MSR)* (May 2013), 233–236.

[4] Graziotin, D. et al. 2014. Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ*. 2, (Mar. 2014), e289. DOI:https://doi.org/10.7717/peerj.289.

[5] Graziotin, D. et al. 2015. How do you feel, developer? An explanatory theory of the impact of affects on programming performance. *PeerJ Computer Science*. 1, (Aug. 2015), e18. DOI:https://doi.org/10.7717/peerj-cs.18.

[6] Guzman, E. et al. 2014. Sentiment analysis of commit comments in GitHub: an empirical study. *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*. (2014), 352–355. DOI:https://doi.org/10.1145/2597073.2597118.

[7] Jockers, M. 2016. R Package "syuzhet."

[8] Jongeling, R. et al. 2017. On negative results when using sentiment analysis tools for software engineering research. *Empirical Software Engineering*. 22, 5 (Oct. 2017), 2543–2584. DOI:https://doi.org/10.1007/s10664-016-9493-x.

[9] Jurado, F. and Rodriguez, P. 2015. Sentiment Analysis in monitoring software development processes: An exploratory case study on GitHub's project issues. *Journal of Systems and Software*. 104, (Jun. 2015), 82–89. DOI:https://doi.org/10.1016/j.jss.2015.02.055.

[10] Jurka, T.P. 2012. R Package "sentiment."

[11] Kalliamvakou, E. et al. 2014. The promises and perils of mining GitHub. *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014* (New York, New York, USA, 2014), 92–101.

[12] Manning, C. et al. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (Stroudsburg, PA, USA, 2014), 55–60.

[13] Mohammad, S.M. and Turney, P.D. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*. 29, 3 (Aug. 2013), 436–465. DOI:https://doi.org/10.1111/j.1467-8640.2012.00460.x.

[14] Murgia, A. et al. 2014. Do developers feel emotions? an exploratory analysis of emotions in software artifacts. *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014* (Hyderabad, IN, 2014), 262–271.

[15] Novielli, N. et al. 2015. The challenges of sentiment detection in the social programmer ecosystem. *Proceedings of the 7th International Workshop on Social Software Engineering - SSE 2015* (New York, New York, USA, 2015), 33–40.

[16] Plutchik, R. 1980. A general psychoevolutionary theory of emotion. *Emotion: Theory, Research and Experience. Theories of Emotion.* H. Kellerman and R. Plutchik, eds. Academic Press Inc. 3–33.

[17] R Core Team 2017. R: A language and environment for statistical computing. R Foundation for Statistical Computing.

[18] Rate Limit: 2017. *https://developer.github.com/v3/rate_limit/*. Accessed: 2017-11-30.

[19] Sinha, V. et al. 2016. Analyzing developer sentiment in commit logs. *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*. (2016), 520–523. DOI:https://doi.org/10.1145/2901739.2903501.

[20] Tourani, P. et al. 2014. Monitoring sentiment in open source mailing lists: exploratory study on the apache ecosystem. *CASCON '14 Proceedings of 24th Annual International Conference on Computer Science and Software Engineering* (Markham, ON, CA, 2014), 34–44.

[21] Vasilescu, B. et al. 2015. Gender and Tenure Diversity in GitHub Teams. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI)*. (2015), 3789–3798. DOI:https://doi.org/10.1145/2702123.2702549.

[22] Vasilescu, B. et al. 2015. Quality and productivity outcomes relating to continuous integration in GitHub. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015* (New York, New York, USA, 2015), 805–816.

[23] Werder, K. 2018. The Evolution of Emotional Displays in Open Source Software Development Teams: An Individual Growth Curve Analysis. *2018 IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering (SEmotion)* (2018), in press.

[24] Xiang, C. et al. 2016. Improving IS development teams' performance during requirement analysis in project—The perspectives from shared mental model and emotional intelligence. *International Journal of Project Management*. 34, 7 (Oct. 2016), 1266–1279. DOI:https://doi.org/10.1016/j.ijproman.2016.06.009.