Proceedings of the 6th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'20)
~~Prague, Czech Republic~~ Virtual Conference – August, 2020
Paper No. EEE 121
DOI: 10.11159/eee20.121

Detecting Emotional Contagion in Collaborative Software Development

Luigi Benedicenti[1]
[1]University of New Brunswick
Faculty of Computer Science
550 Windsor Street, Fredericton, NB, Canada
Luigi.Benedicenti@unb.ca

Abstract **–** In this paper, a method is presented to detect emotional contagion in collaborative software development where such collaboration happens through online means, and more specifically GitHub. The method is based on a previously published framework to classify interactions among software developers in open source software collaborations. This paper builds on that framework to provide a method that can effectively detect and classify the extent to which emotional contagion appears to be present in a specific development effort. The method is general and can be applied to any development that uses GitHub as primary means of communication. By using a popular open source software project, this paper shows that within the selected scope it is possible to detect emotional contagion. The validity of the result is confined to each project, as a generalization of this analysis is beyond the scope of the paper, but is described as future research.

Keywords**:** Emotional Contagion, Software Engineering, Open Software, Affect Theory.

1. Introduction

Software Engineering is a human-centred process driven by creativity and collaboration [1, 2, 3]. As human behaviour is influenced by emotion, the application of Affect Theory to Software Engineering is increasingly relevant. Although a great deal of work has been conducted on the subject, some aspects of the collaborative nature of the software development process are just beginning to be explored. One such aspect is Emotional Contagion.

Emotional Contagion is the manner in which an affect demonstrated by one actor is transmitted to other actors. This occurs when the manifestation of a certain behaviour generates a similar, reactive behaviour in a person or a group. For example, a software developer expresses a severely negative opinion on a piece of code in an environment that encourages individual code ownership, thus triggering a defensive reaction in the code owner that results in a heated exchange that may involve the entire development group [4].

This paper describes a systematic method to detect emotional contagion in a collaborative software project whose primary interaction among project contributors occurs via an online repository, in this particular case GitHub. This method is based on a previously published framework [5]. The work presented in this paper is useful to create a reference for the characterization of emotional contagion in collaborative software development, and may lead to a measure of the impact of emotional contagion on team productivity, software quality, and development time.

2. Previous Work

Research on Affect Theory applied to Software Engineering is growing [1, 2, 3]. The state of the art attempts to characterize the influence of affect on software production while adopting different perspectives. Overall, research work has concentrated on the correlation between affect and productivity or quality, and then this has encompassed a more holistic approach that comprises attributes of the development process, qualities of the artefacts produced, and well-being of the participant in the process [4, 5].

Some of the more recent focus of such research is in the agile requirements field [6]. Part of the author's own research has focused on optimization of requirements prioritization methods using decision methods to reduce the occurrence of emotional contagion [7]. In such a knowledge-intensive activity, cognitive trust plays an important role, but so does affective trust, which extends to the social perspective of the concept of trust [3].

When distributed development occurs, there can be many types of interaction, even if in-person interaction is limited. The type of interaction varies depending on many factors, among which are whether or not the developers work as part of a corporate structure, the technical resources available to each developer, the infrastructure that each developer can afford, and many others. In this paper, we will concentrate only on interaction that occurs asynchronously over a single point of contact: a repository. This limits the validity of the paper but makes it possible to analyse all the interactions as they are automatically documented.

The constraints imposed by repository-based online interactions naturally limit the opportunities for the development of cognitive trust, since it is relying on a single source at a time (i.e., the developer's check in message). However, given that that source comes with a number of artefacts, there is a way for every contributor to assay the cognitive trust of every other contributor. On the other hand, affective trust can only be based on the tone of comments in the repository. The limited information contained in the comments, therefore, makes it possible to generate emotional contagion.

3. Method Design and Results

The method described in this paper works under the same assumptions as the previously published framework [5]. For the sake of convenience, a short summary of the framework is included below. The framework assumes that limited or no face-to-face interactions occur among developers, and that a single version control system is used to exchange information and system artefacts. In our work, we have concentrated on Git and Github as the version control system and the repository for artefacts respectively [8, 9].

The framework includes for steps for the analysis of interactions among developers. They are as follows:

1. Retrieve data from a repository
2. Sort it by locality
3. Evaluate the affective content of the data
4. Determine if episodes of contagion exist

The first step is achieved by using the repository commands for retrieving all the information on every modification stored in the repository. In our work, we have used Git [8, 9]. In this case, the information we retrieve is the change log and the commits notes. The information we collect is then processed so as to be loadable by Mathematica, the tool we have adopted for our analysis [10].

The second step is achieved by creating two sets of data. In the first set, data is sorted by commit time only. This reflects one locality aspect of emotional contagion: the immediacy of reaction. This set is very easy to process, and we will use it in the example that follows. The second set is data binned by affected artefact (for Git, this is a source file) and then sorted by time. This set requires a more sophisticated processing step, but it is useful for counting reactions to changes related to a specific code unit (e.g., a class, protocol, or function). A third type of locality could be introduced, as more structured projects also reference issue numbers. However, at this time, since fewer projects adopt issue numbers, we have refrained from making use of it.

The third step is achieved by performing a sentiment analysis on the two datasets. The tool we chose to use, Mathematica, contains a recent and reasonably precise sentiment analysis tool, which we have adopted for our examples. More precise and targeted sentiment analysis tools exist, and in future work we will consider comparing the results obtained with Mathematica with other sentiment analysis tools. However, since Mathematica is highly versatile and provides much more than the sentiment analysis tool, at this time its use is considered reasonable.

The final step is to determine if episodes of contagion exist. In our previous work, the analysis was done mostly visually, by assessing trends in a plot of the integrated dataset. Although this is a good qualitative measure of the presence of emotional contagion, it does not allow for a quantitative analysis. The method proposed below provides such quantitative evaluation. For comparison, we include a picture of a qualitative analysis below, showing emotional contagion as peaks and valleys of the curve shown in the picture.

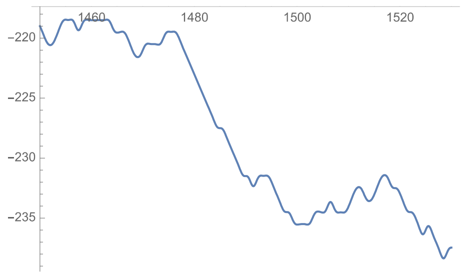


Fig. 1: Qualitative analysis of emotional contagion.

The method described in this paper is instead based on pattern recognition. Emotional contagion is defined as a series of identical reactions that change a previous trend. For example, if the reactions were univocally negative and then a positive reaction started a chain of positive reactions, that is the result of positive emotional contagion. Similarly, if reactions were mixed, mostly neutral, and then a negative reaction is followed by a sequence of negative reactions, that is the result of negative emotional contagion. This is relatively easy to compute in Mathematica, which has native representation and computation of patterns. However, one needs to be careful in defining the problem as unfortunately the general solution is computationally intensive.

We will now describe the quantitative method through an example. We have chosen the same GitHub project we analysed qualitatively [5], to ensure continuity among the results published. The project we chose for our example is Opencv, a large open source software project [11].

Figure 1 shows the results of the qualitative analysis. This analysis was performed on a list sorted by time only, adopting the first locality principle for simplicity. At the end of step 3 of the analysis, we obtained two datasets, one resulting from the chronological analysis of the short comments in the project, and the other resulting from the chronological analysis of the long comments in the project commits. In the dataset, a positive value indicates a positive comment; a value of zero indicates a neutral comment; and a negative value indicates a negative comment. The format of these datasets is shown below in Fig. 2.

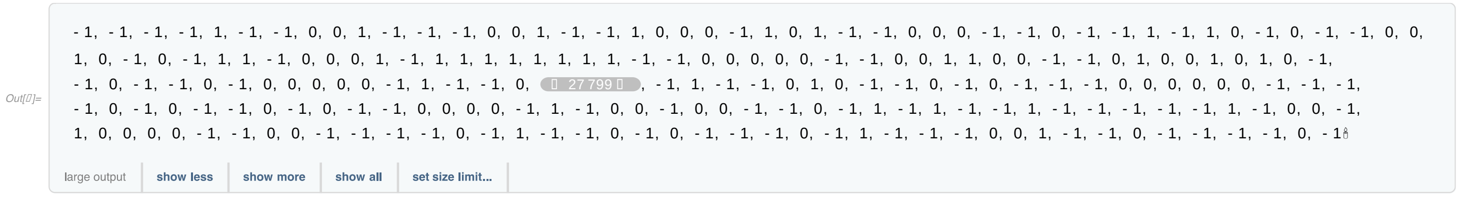
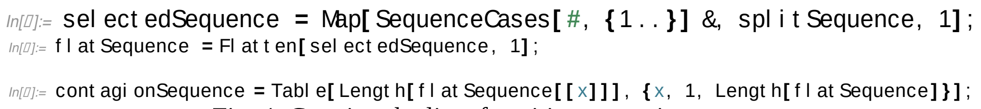

```
Out[]=
- 1, - 1, - 1, - 1, 1, - 1, - 1, 0, 0, 1, - 1, - 1, - 1, 0, 0, 1, - 1, - 1, 1, 0, 0, 0, - 1, 1, 0, 1, - 1, - 1, 0, 0, 0, - 1, - 1, 0, - 1, - 1, 1, - 1, 1, 0, - 1, 0, - 1, - 1, 0, 0,
1, 0, - 1, 0, - 1, 1, 1, - 1, 0, 0, 0, 1, - 1, 1, 1, 1, 1, 1, 1, 1, 1, - 1, - 1, 0, 0, 0, 0, 0, - 1, - 1, 0, 0, 1, 1, 0, 0, - 1, - 1, 0, 1, 0, 0, 1, 0, 1, 0, - 1,
- 1, 0, - 1, - 1, 0, - 1, 0, 0, 0, 0, 0, - 1, 1, - 1, - 1, 0,  27 799 , - 1, 1, - 1, - 1, 0, 1, 0, - 1, - 1, 0, - 1, 0, - 1, - 1, - 1, 0, 0, 0, 0, 0, 0, - 1, - 1, - 1,
- 1, 0, - 1, 0, - 1, - 1, 0, - 1, 0, - 1, - 1, 0, 0, 0, 0, - 1, 1, - 1, 0, 0, - 1, 0, 0, - 1, - 1, 0, - 1, 1, - 1, 1, - 1, - 1, 1, - 1, - 1, - 1, - 1, - 1, 1, - 1, 0, 0, - 1,
1, 0, 0, 0, 0, - 1, - 1, 0, 0, - 1, - 1, - 1, - 1, 0, - 1, 1, - 1, - 1, 0, - 1, 0, - 1, - 1, - 1, 0, - 1, 1, - 1, - 1, - 1, 0, 0, 1, - 1, - 1, 0, - 1, - 1, - 1, - 1, 0, - 1
```

Fig. 2: Example dataset format (short comments).

Using Mathematica's own built-in functions, it would be very simple to conduct an analysis of the dataset above to detect the trends indicating emotional contagion. For example, to detect positive emotional contagion we could write the following command.

```
In[]:= Map[Length, SortBy[SequenceCases[shortCommentSequence, {1..}], count]]
```
Fig. 3: One-step analysis command.

Unfortunately, the computational complexity of this algorithm proves to be too high for the number of points in the dataset, which is 28,017. As the complexity of this algorithm is estimated to be of the order of the cube of the number of data points, this calculation would take around 5,000 years to complete. Luckily, there are faster ways to achieve the same result. In essence, we ask Mathematica to split the entire sequence in a series of short sequences, each of which is a contagion event. Then, we count the number of elements in each list. The result is a list of events in which contagion exists.



```
In[]:= selectedSequence = Map[SequenceCases[#, {1 ..}] &, splitSequence, 1];
In[]:= flatSequence = Flatten[selectedSequence, 1];

In[]:= contagionSequence = Table[Length[flatSequence[[x]]], {x, 1, Length[flatSequence]}];
```

Fig. 4: Creating the list of positive contagion events.

It is now relatively straightforward to create a frequency table of the occurrences, and the result is shown below.

Table 1: Number of occurrences per length (positive contagion).

Sequence Length	Number of times occurring
2	426
3	1
4	89
5	33
6	1
8	2
10	5
13	12
49	1

This shows that the quantitative analysis is possible for both types of emotional contagion, as the algorithm described is extremely versatile. In this specific example, we can see that positive contagion appears for more than just 2 comments, which suggests that in fact emotional contagion is indeed happening. If we ignore the short 2-element sequences, the distribution of the remaining occurrences seems to suggest a bimodal curve with two peaks, one around 4 and 5 elements, and one around 13 elements. This seems to suggest that in this particular case, emotional contagion can be a short-timed event or a longer event, with a preponderance of short-timed events. Whether this is the result of a highly disciplined group or other factors is a topic for future work.

4. Conclusions

This paper has shown a method to quantitatively detect emotional contagion in a sequence of comments from commits in a GitHub project. This method is versatile, expandable, and computationally light. It uses repeatable algorithms that can be applied to every GitHub project, and with some additional work, other repositories. The emotional contagion information can be analysed and validated, although external validation is outside the scope of this paper.

The paper has shown only one example, centred on positive emotional contagion in a single project, that however counts more than 28,000 elements. Moreover, the time locality principle has been shown, which is only the simpler of three locality principles that can be adopted for the analysis.

In the future, this method will be applied to a larger number of datasets, and the findings will be compared to determine whether or not a pattern can be discerned. Furthermore, the amount of emotional contagion will be used to determine if there is a relationship between emotional contagion and development process in terms of productivity and quality of the artefacts produced.

References

[1] Graziotin, D.: Towards a Theory of Affect and Software Developers' Performance. PhD Dissertation as defended on January 12, 2016 at the Faculty of Computer Science of the Free University of Bozen-Bolzano.

[2] Graziotin, D., Fagerholm, F., Wang, X., Abrahamsson, P.: On the Unhappiness of Software Developers. In: Proceedings of the 21st International Conference on Evalua-tion and Assessment in Software Engineering (EASE'17), 2017. ACM, New York, NY, USA, 324-333.
[3] Calefato, F., and Lanubile, F.: Affective trust as a predictor of successful collaboration in distributed software projects. In: Emotional Awareness in Software Engineering (SEmotion), IEEE/ACM International Workshop. IEEE, 2016, pp. 3–5.
[4] Graziotin, D., Fagerholm, F., Wang, X., Abrahamsson, P.: What happens when soft-ware developers are (un)happy. Journal of Systems and Software. Volume 140, June 2018, Pages 32-47.
[5] Benedicenti, L.: Emotional Contagion in Open Software Collaborations. In: Ivanov V., Kruglov A., Masyagin S., Sillitti A., Succi G. (eds) Open Source Systems. OSS 2020. IFIP Advances in Information and Communication Technology, vol 582. Springer, Cham, 2020.
[6] Ochodek, M., and Kopczynska, S.: Perceived importance of agile requirements engineering practices–a survey. Journal of Systems and Software, 2018.
[7] Alhubaishy, A., and Benedicenti, L.: Toward a model of emotion influences on agile decision making. In: Proceedings of the 2nd International Workshop on Emotion Awareness in Software Engineering. IEEE Press, 2017, pp. 48–51.
[8] GitHub homepage, http://www.github.com, last accessed 2020/05/02.
[9] Git homepage, http://git-scm.com, last accessed 2020/05/02.
[10] Wolfram Mathematica homepage, http://www.wolfram.com, last accessed 2020/05/02.
[11] Opencv GitHub page, https://github.com/opencv/opencv, last accessed 2020/05/02.