Emotional Contagion in Open Software Collaborations

Luigi Benedicenti

Emotional Contagion in Open Software Collaborations

Luigi Benedicenti[1][0000-0003-4036-9593]

[1] University of New Brunswick, Fredericton NB E3B 5A3, Canada
`luigi.benedicenti@unb.ca`

Abstract. Emotional contagion is a mechanism by which affect experienced by one person in a group is transmitted to others in the same group. When this happens, the group dynamic is influenced. This paper provides a method to analyze an Open Software project to determine the connection between emotional contagion and software production in such an environment, if any. The project change management database is mined to extract change comments in chronological order and by user id. Sentiment analysis is employed to determine affect in the change originating from each userid. File changes are tracked to link them together in the same areas, using a temporal and file locality principle. The correlation between affect and area is then used to prove or disprove whether or not emotional contagion influences open software production. Although in this paper the proposed method is applied to only one project, the method is general and can be reused for experimental validation.

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

1 Introduction

The application of Affect Theory to Software Engineering is becoming relevant in modern software development that involves collaboration [1, 2, 3]. This is particularly relevant in environments in which collaboration occurs remotely, because it is a very well-known fact that online comments allow and sometimes encourage a less respectful engagement. Open Source Systems (OSS) are a particularly apt instance of this trend.

One particularly relevant aspect of Affect Theory in the case of OSS is Emotional Contagion. Emotional Contagion occurs when the affect inherent in a behavior, or in the case of online collaboration a message, is shared with others. The emotional content of the message can alter the affect associated with people who read the message, enacting a propagation of affect that resembles the spread of a contagion [4].

This paper describes a systematic method to analyze an OSS project to determine whether emotional contagion occurs. This method is relevant to validate the hypothesis that emotional contagion occurs in OSS projects, and to determine the point in time when this happens, to create the opportunity to assess how the emotional contagion may be affecting the development process and the quality of its output. The method presented in this paper is sufficiently general to be repeatable, allowing for comparisons and meta-analyses that can quantify the degree of emotional contagion occurrence in OSS and support future analyses and the creation of new development processes.

2 Previous Work

There is a growing body of work on Affect Theory applied to Software Engineering [1, 2, 3]. The research in this area is an attempt to characterize the influence of affect on software production from different points of view. Although initially the work has concentrated on the correlation between affect and productivity or quality, gradually this focus has been expanded to a more holistic approach that involves attributes of the development process, qualities of the artifacts produced, and well-being of the participant in the process [4].

Some of the more recent focus of such research is in the agile requirements field [5]. 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 [6]. In such a knowledge-intensive activity, cognitive trust plays an important role, but so does affective trust, which encompasses the social perspective of trust [3].

In distributed OSS development, interactions are more severely limited, as in most cases opportunities for face-to-face meetings and teleconferences are non-existent. All that remains is, at best, a combination of social networks, email lists, and comments in software repositories. In most cases, in fact, software repositories are all that links together each developer.

The constraints imposed by repository-based online interactions in turn limit the opportunities for cognitive trust, as the information on contributors only comes from a single source. However, given that the source contains a number of artifacts and documented interventions, cognitive trust is not severely impaired. On the other hand, affective trust can only be based on the tone of comments in the repository. This limited amount of information can lead to misunderstandings, creating an environment that is ripe for emotional contagion.

3 Method Workflow

The development of a method to analyze the influence of affect on OSS development, and in particular the presence of emotional contagion, is grounded in a series of principles and assumptions. These principles and assumptions help define the scope of the method, which is important in determining its applicability.

The method presented in this paper works under the assumption that face-to-face interactions among developers are severely limited or non-existent. This assumption is paramount to the definition of the data set for the analysis. Therefore, the applicability of this method on teams in, for example, a company, is not considered.

This method also relies on the locality principle, in that the influence of affect in standard working conditions is mitigated by time and by the location of each change. In other words, recent changes to a common artifact have higher impact than changes distributed over several days and artifacts. This is because most emotions and moods change relatively rapidly (over a period of hours) and the sense of ownership for a specific artifact lowers as the developer's focus shifts.

A further assumption is that there exists a repository that contains a documented evolution of the software being produced. An example of such a repository is GitHub [7].

Given these assumptions and principles, it appears reasonable to limit the scope of the method to situations that do not allow direct face-to-face interaction and that involve a relatively large number of developers. OSS development falls within this scope, and is what this paper focuses on.

The method's workflow, therefore, is 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

Each of these steps is described below. To illustrate the method, we will use an example that will be further discussed in the next section.

3.1 Retrieve Data from a Repository

Data retrieval from a repository is a relatively straightforward operation. To keep things as simple as possible, the retrieval makes use of generally available tools like a text editor and a spreadsheet, and adopts general file formats like text and comma-separated values. Even with these simplifications, the data retrieval procedure remains delicate and needs to be checked for accuracy.

The requirements for the choice of a repository is that it contain the information needed to perform an analysis of the affect of a contribution, in a chronological order, and with an indication of the artifacts that have been changed. At a minimum, therefore, the data needed for the analysis is the date of the contribution, the file(s) affected by the contribution, and a comment explaining the contributions made.

Our primary choice for a repository is GitHub [7]. This repository contains all the needed information, is free for most open software development, and it is very popular, although it might not be considered the most advanced software repository currently available. Another advantage is that the software program that allows interaction with the repository, Git, is a free and open source software available on most platforms, has been proven extensively, and is supported by a large community of developers [8].

To retrieve the data from the repository, we first clone it on the local machine. After that, we extract the information we need using Git commands. In particular, the log command can be used to obtain all change information with the exception of the files affected. For that, we need to use the diff command. Depending on the next steps, it might be necessary to further process the data into a format that is readable by the analytical tool. In our case, we have processed the data to create a comma-separated value list that is readable by Wolfram Mathematica, our choice of analytical too [9].

3.2 Sort It by Locality

The sorting by locality is accomplished in two ways. The sorting by date is simple, as we can access the change log and sort it accordingly. The sorting by file is a bit more complex, as a change item can encompass multiple files. In this case, we resort to creating an entry for each file, and then sorting them into separate bins, one per file. This results in a set of bins, one per file, each of which is sorted by date.

3.3 Evaluate the Affective Content of the Data

To evaluate the affective content of the data we make use of a sentiment analysis classifier. Because Mathematica provides an existing classifier, we adopted it, keeping in mind that this built-in classifier only works for English words.

Sentiment Analysis is an effective tool for the determination of affective content, but it has its limitations. Firstly, the classifier comes with a customizable level of confidence. Secondly, it is based on a machine learning algorithm that depends on the level of training that the classifier has received. Thirdly, sentiment analysis only provides an indication of the type of affect in a sentence (Positive, Neutral, or Negative); but not the intensity of the affect.

This can be problematic in many ways. The customizable level of confidence needs to be declared in any analysis to make it repeatable. Further, the use of a preset classifier means that it is not possible to control the type of training the classifier has received. Our reliance on a general-purpose sentiment analysis tool is a restriction of the usefulness of the method, because to make this repeatable we need to ensure that the same classifier is used on every data set.

Additionally, the classifier result is discrete. The provision of an intensity value for sentiment analysis is not available in most tools, which limits the level of refinement of the analysis.

3.4 Determine If Episodes of Contagion Exist

Numerous options exist for this determination. The simplest of these options is to generate a series of affect sequences, which are chronological representations of the changes classified by the sentiment analysis tool for a specific file, and then perform a manual inspection of each generated affect sequence.

Other possibilities include more sophisticated analyses, which involve a pattern recognition algorithm to detect a change in affect following a single change comment (i.e., influencing), or even convolutional analysis with kernels designed to highlight emotional contagion.

Our implementation of this stage is to translate the results of the affect sequence into numerical data: -1, 0, and 1 for positive, neutral and negative values respectively, then integrate these numerical sequences over time, and provide a graphical representation that can be further analyzed.

It is important to note, however, that the preceding data collection and processing is fully repeatable, and that should additional algorithms become available, it would be possible to apply the new algorithms to the same data sets without any loss of generality.

4 Example

The simple example we present in this section exemplifies the steps presented in the previous section. The example has been chosen purposely to make it as simple as possible, rather than as comprehensive as possible. Thus, this example has limited external validity.

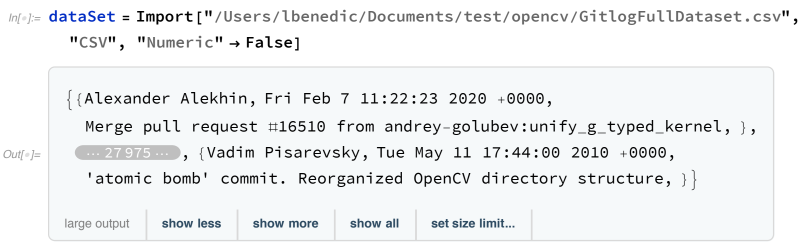
Github has a large number of OSS projects. A small but representative one is opencv [10]. This open source computer vision library has received contributions from more than 1,000 contributors and has a change log with more than 27,000 entries. Although not the largest contribution by any means, it is sizeable enough to prove interesting for this example.

To acquire the original data set, we cloned the GitHub library in a local directory on the research machine (an iMac Pro with 64GB of RAM and a 10-core Intel Xeon W processor). We then extracted the information we needed from the repository with a combination of Git commands, an example of which is below (see Fig. 1).

```
git clone https://github.com/opencv/opencv
git log --pretty=format:'"%an","%ad","%s","%b"' > ./GitlogFullDataset.csv
```

Fig. 1. GitHub commands for data set extraction (Sample).

Following the extraction, we prepared the data for ingestion by Mathematica, and then loaded it (see Fig. 2).

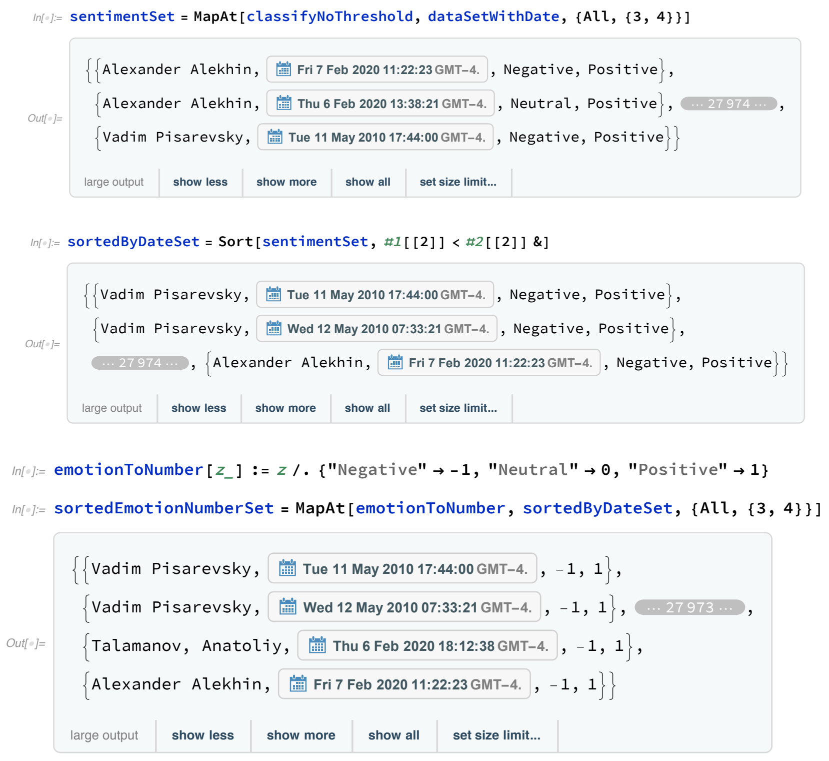


```
In[∘]:= dataSet = Import["/Users/lbenedic/Documents/test/opencv/GitlogFullDataset.csv",
          "CSV", "Numeric" → False]

Out[∘]= {{Alexander Alekhin, Fri Feb 7 11:22:23 2020 +0000,
          Merge pull request ⌗16510 from andrey-golubev:unify_g_typed_kernel, },
         ⋯ 27975 ⋯, {Vadim Pisarevsky, Tue May 11 17:44:00 2010 +0000,
          'atomic bomb' commit. Reorganized OpenCV directory structure, }}
```

Fig. 2. Importing the data into Mathematica.

Sentiment analysis and locality can be applied very simply, when the data is already binned; it is then possible to create an affect sequence (see Fig. 3).



```
In[•]:= sentimentSet = MapAt[classifyNoThreshold, dataSetWithDate, {All, {3, 4}}]

Out[•]= {{Alexander Alekhin, Fri 7 Feb 2020 11:22:23 GMT-4. , Negative, Positive},
         {Alexander Alekhin, Thu 6 Feb 2020 13:38:21 GMT-4. , Neutral, Positive}, ··· 27 974 ··· ,
         {Vadim Pisarevsky, Tue 11 May 2010 17:44:00 GMT-4. , Negative, Positive}}

         large output   show less   show more   show all   set size limit...

In[•]:= sortedByDateSet = Sort[sentimentSet, #1[[2]] < #2[[2]] &]

Out[•]= {{Vadim Pisarevsky, Tue 11 May 2010 17:44:00 GMT-4. , Negative, Positive},
         {Vadim Pisarevsky, Wed 12 May 2010 07:33:21 GMT-4. , Negative, Positive},
         ··· 27 974 ··· , {Alexander Alekhin, Fri 7 Feb 2020 11:22:23 GMT-4. , Negative, Positive}}

         large output   show less   show more   show all   set size limit...

In[•]:= emotionToNumber[z_] := z /. {"Negative" → -1, "Neutral" → 0, "Positive" → 1}

In[•]:= sortedEmotionNumberSet = MapAt[emotionToNumber, sortedByDateSet, {All, {3, 4}}]

Out[•]= {{Vadim Pisarevsky, Tue 11 May 2010 17:44:00 GMT-4. , -1, 1},
         {Vadim Pisarevsky, Wed 12 May 2010 07:33:21 GMT-4. , -1, 1}, ··· 27 973 ··· ,
         {Talamanov, Anatoliy, Thu 6 Feb 2020 18:12:38 GMT-4. , -1, 1},
         {Alexander Alekhin, Fri 7 Feb 2020 11:22:23 GMT-4. , -1, 1}}

         large output   show less   show more   show all   set size limit...
```

Fig. 3. Sentiment analysis, sorting, and affect sequence creation.

The resulting sequence can be integrated and plotted. In Fig. 4, we show the first 500 data points in the affect sequence and highlight a few instances of emotional contagion found in it. The X axis is time, and the Y axis is the emotional accumulation level.

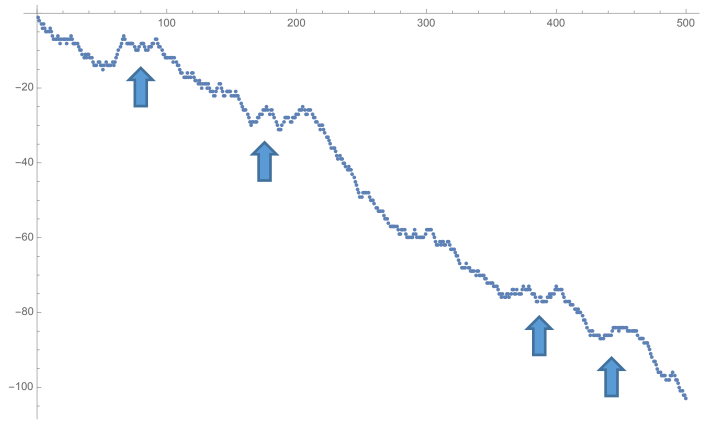


Fig. 4. Affect sequence plot.

5 Discussion

There are a number of restrictions that come from the adoption of principles and assumptions detailed in Section 3. These restrictions affect the applicability of the method and its validity.

In terms of applicability, not all repositories are suitable for this kind of analysis. Repositories where work is checked in by a very small group of developers are unsuitable for analysis because the assumption of low cognitive and affective trust is not true. In general, small communities are able to organize much more tightly, which leads to higher levels of cognitive and especially affective trust. This may happen in larger communities too (consider the level of affective trust offered to longstanding community contributors such as Linus Torvalds, for example), but it is much rarer.

As well, repositories where contributors all work in close proximity will not be suitable as the increased level of communication, especially for face-to-face communications, increases the level of affective trust. If this higher level of affective trust is verified, however, these repositories may be used to detect whether a higher level of affective trust changes the occurrences of emotional contagion.

In terms of validity, internal validity greatly depends on the availability of a large number of check-in information, and the ability of the developers to provide clear commentary with some sort of emotional content (either explicit or implied).

The more difficult form of validity, however, is external validity. Results from a single data set have no external validity and can only be representative of the repository they come from. To obtain a degree of external validity, many repetitions will be needed and a common format for the presentation and archival of results will be necessary. If this happens, then a distribution of results will be created that can be representative of a category of software development falling within the scope of this method.

6 Conclusions

This paper presented a method to determine whether emotional contagion occurs in OSS development. The method relies on the locality principle, and assumes that in-person interactions are limited, and that contributors rely on the comments in the repository to coordinate their work. The method works by analyzing the emotional content of developers comments in code check-ins to repositories. The example provided shows that the method is able to detect patterns in emotional content showing emotional contagion in a specific repository.

The method is limited in its scope and validity by the assumptions made in developing it. As well, its external validity cannot be assessed through a single example. Future work includes further streamlining of the data collection, a modified analysis method that relies on interpolated points to perform convolutional analyses in addition to the standard visual inspection, and a better structured manner to integrate the results from each local file bin.

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 Evaluation 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 software developers are (un)happy. Journal of Systems and Software. Volume 140, June 2018, Pages 32-47.
5. Ochodek, M., and Kopczynska, S.: Perceived importance of agile requirements engineering practices–a survey. Journal of Systems and Software, 2018.
6. 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.
7. GitHub homepage, http://www.github.com, last accessed 2020/01/02.
8. Git homepage, http://git-scm.com, last accessed 2020/01/02.
9. Wolfram Mathematica homepage, http://www.wolfram.com, last accessed 2020/01/02.
10. Opencv GitHub page, https://github.com/opencv/opencv, last accessed 2020/01/02.