Social Network Analysis in GitHub - A Study of Repositories Similarities

Rida Boushab,
Vasconselos Oliveira
High School of Computer Sciences, Mohammed V University

May 19, 2020

Abstract

Social coding platforms has changed the way programmers create and maintain open sources software projects, and GitHub has become the de facto social coding platform. Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
In this paper, we contribute to the body of knowledge of this social coding platform by modeling and studying the similarities between repositories. And we propose an algorithm to make a developer recommendation system for projects.

1 Introduction

In our days, many developers has at least once, used a social coding platform, such as GitHub [3]. These platforms allow developers to experience new ways of broadcast their activities and/or listen to activities of others; Since users are always searching the collaborators for theirs projects or the projects to collaborate, it's important to model repositories and analyse the similarities between them in order to recommend a right developer to a right project. Such studies are important as they help to improve the collaborative work by recommending developers with common background and interest. By analyzing the projects similarity and users interest we can build a recommendation system and make a link prediction between users.

2 Methodology

In this section we describe our methodology to construct a sample network from GitHub repositories, the model to use with Neo4j database and how we measure the similarity of GitHub repositories.

2.1 Graph Data Modeling

As we want to analyse our social coding platform, we decided to model it as graph, since modeling the social platform as graph allow us to better capture its characteristics and relationship.

Let R denote a project repository and U denotes a user of github, that we consider as nodes, containing all metadata respectively. For R, the corresponding metadata are: a list of branches, a list of collaborators, a list of comments, a list of commits, a JSON file describing the community, a JSON file describing the project contents, a list of deploy keys, a list of deployments, a list of downloads, a list of forks, a list of invitations, list of all the releases of the repository, the statistics of the repository (such as contributors list with additions, deletions, and commit counts) and the list of commit status.

For U the respective metadata are: the id, the login, the list of followers and the list of who the user follows, and the list of repositories.

Our nodes are linked by edges which describe the relation between them. In this project we choose the following relationship, as shown in figure 1:

- **Is the Owner of**: describes the relationship of belonging;
- **Is a Collaborator in**: describe the relation of collaboration in a project.

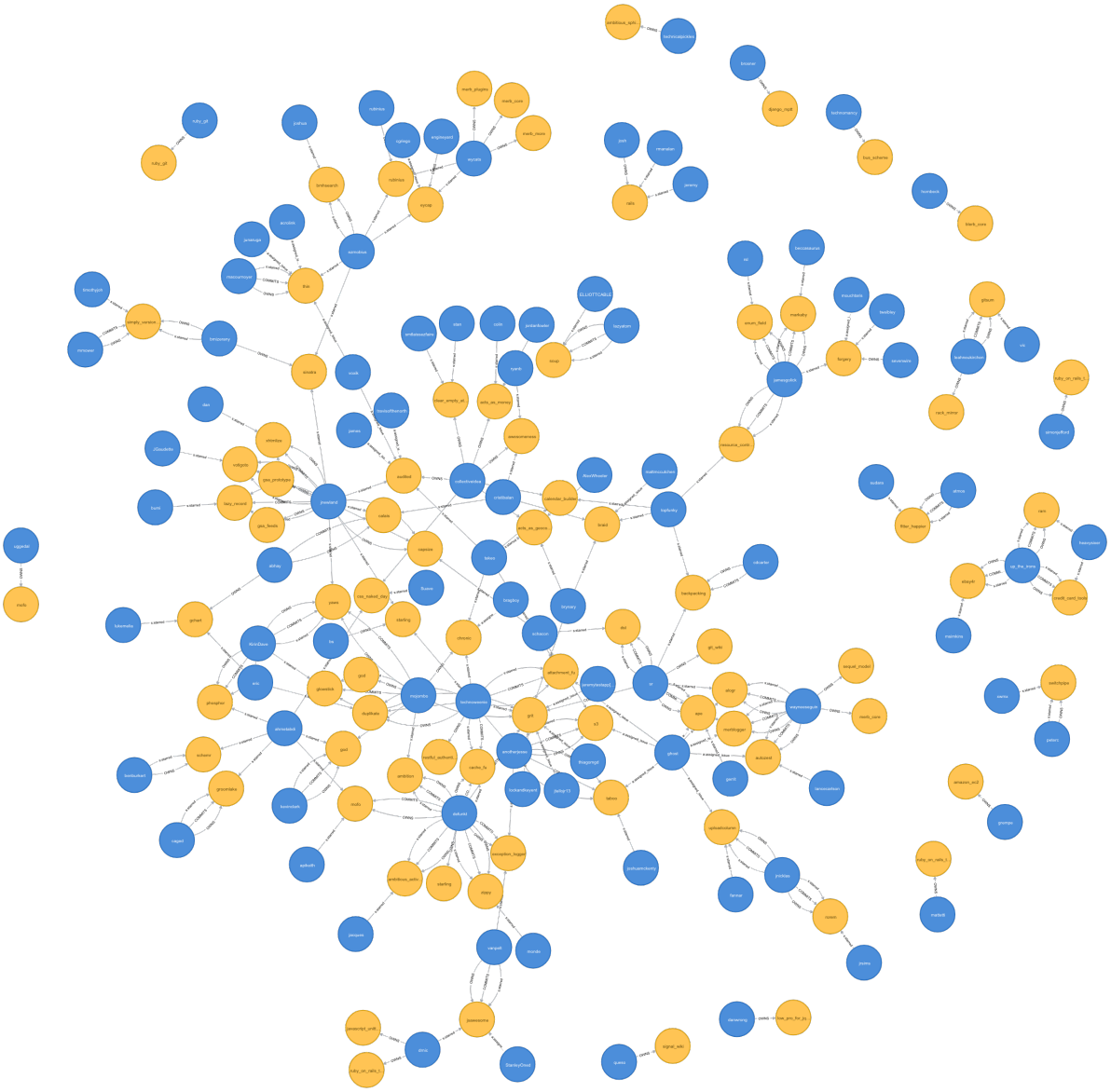

Figure 1: GitHub User-Repository graph model

2.2 Similarity

In this section, we describe how we are going to measure the similarity of repositories on GitHub.

2.3 Concept

Finding similar repositories on GitHub can be helpful for software engineers as it can help them reuse source code, identify alternative implementations, explore related projects, find projects to contribute to, and discover code theft and plagiarism. Let's take this example: U1(user 1) has the repository C1, another user U2 has a repository C2 and the user U3 has a repositort C3.

A.the Star concept: The Projects that are starred by the same users are likely to be similar with one another
B. User programming language: If the user U2 has one of the same programmig language in the repository of the user U3, so there's a small probanilty that they gonna have the same repository.
C.Projects whose readme files contain similar contents are likely to be similar with one another: by using Text mining we can model the topic of the readme file to see if they are similar or not.
D.Transition committer: If user U2 commit in the repository of the user U1 and the the User U3 commit in the repository U2, so by transition U3 can have similar repository with U1.
E. Issues section: if the user U2 report an issue about the repository of the user U1, and the category of the U2 should be an "enhancement", so the repository between U1 and U2 could be similar.

2.4 Architecture

In this section, we are going to define a formula to compute the similarity score between two repositories.

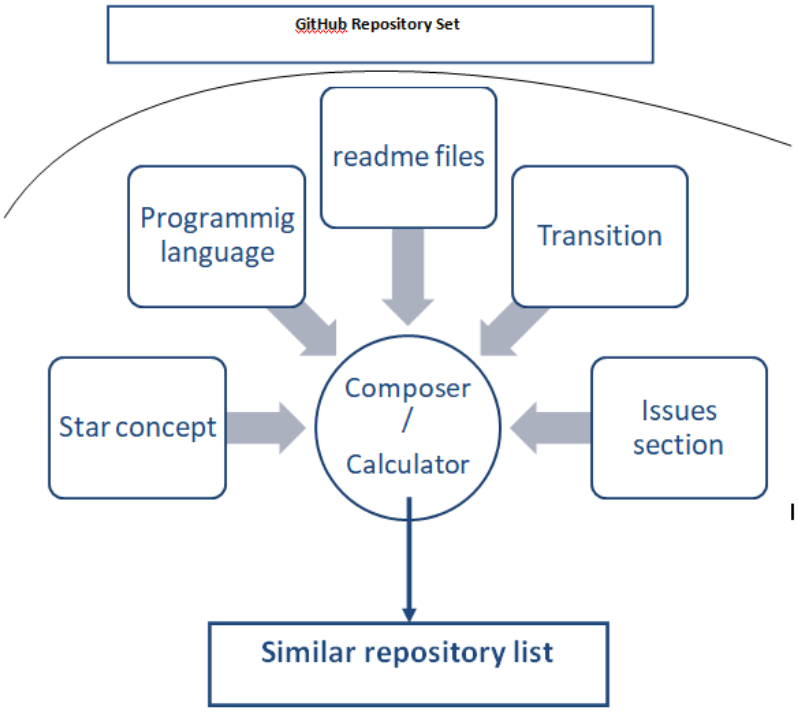

Figure 2: Caption

2.5 Star formula

two repositories that are starred at a similar period of time by many people are likely to be similar together. lets use this intuition to calculate the star formula:

t = $|T(U, R1) - T(U, R2)|$

StarScore= $\begin{cases} 1 \quad if \quad t \in [0, 10] \\ 1 - \frac{t-10}{100} \quad if \quad t > 10 \end{cases}$ $*$

such that :

t is the difrrence time of two repositories that are starred in minute

T is the time that the user star a repositorie

$*$ if the user is stared the two repositories R1 and R2 between 0 to 10 minutes that's the top score of the star score.

2.6 Readme formula

We compute this relevance score using the Cosine Similarity, which is commonly used to find similarity between documents in information retrieval.after calculating the tf-idf weighting(WR), we gonna use the cosine similarity to calculate the similarity between R1, R2:

$$\left[ReadmeScore = \frac{WR1 * WR2}{\|WR1\| * \|WR2\|} \right]$$

2.7 Transition formula

If user U2 commit in the repository of the user U1 and the the User U3 commit in the repository U2, so by transition U3 there's a chance of having similar repository with U1.

$$\text{TranScore}= \begin{cases} 1 & if \quad Commit \\ 0 & if \quad Not\ commit \end{cases}$$

2.8 Issue formula

if the user U2 report an issue about the repository of the user U1, and the category of the U2 should be an "enhancement", so there's a chance of having similar repository. we calculate the score like that:

$$\text{IssueScore}= \begin{cases} 1 & if \quad Similar \\ 0 & if \quad NotSimilar \end{cases}$$

2.9 Language formula

$$\text{LanguageScore}= \begin{cases} 1 & if \quad Same\ Languagec \\ 0 & if \quad Not \end{cases}$$

2.10 Composer algorithm

INPUT = R2, R1
OUTPUT =

$$\left[composer = \frac{Readme(R1, R2) * 50 + star(R1, R2) * 30 + Issue(R1, R2) * 20}{100}\right]$$

if composer >= 0.5 then
return R1 AND R2 ARE SIMILAR
ELSE return Not SIMILAR
end

3 README Files Scores

In order to compute the similarity of README files among repertories, techniques of text mining were used, such as **Term Frequency-Inverse Document Frequency (TF-IDF)** and the well known metric used to measure how similar the documents are, **the Cosine Similarity**.

3.1 Term Frequency-Inverse Document Frequency (TF-IDF)

Intuitively, **tf-idf** provides a means of characterizing/describing documents using the words that occur in them. It provides a means for converting documents into Vectors [N1, N2, N3....Nn] for machine learning or further processing. Given a collection of documents that make up a corpus, tf-idf assigns scores to words in each document which represents how important the word is in describing the document. 'Generally', the more important the word, the higher it's tf-idf score. The score is a product of two distinct terms TF (term frequency - how often the word occurs in the document) and IDF (Inverse Document Frequency - is a weighting factor that values rare words. It is constant for each word in the corpus and is a measure of how rare the word is in the corpus.)[1]

So what we are trying to do with tf-idf is to find a vector that best describes the document using the words that occur in the document, giving more value to words that are rare in the corpus.

3.1.1 Term frequency

$$TF - IDF = TF_t \times \log \frac{N}{DF_t} \tag{1}$$

TF_t: Term Frequency of the term t. How many times does the term occur in the document.

N- Total number of documents in the corpus.

DF_t: Document Frequency of t. How many documents have the term t.

A number of approaches exist for computing TF-IDF including normalizing the term frequencies to account for the variation in length of the documents in the corpus.[1]

3.2 Cosine Similarity

Cosine similarity is a metric used to determine how similar the documents are irrespective of their size.

Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. In this context, the two vectors I am talking about are arrays containing the word counts of two documents.[2]

When plotted on a multi-dimensional space, where each dimension corresponds to a word in the document, the cosine similarity captures the orientation (the angle) of the documents and not the magnitude.

The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance because of the size they could still have a smaller angle between them. Smaller the angle, higher the similarity.citeinproceedings

$$\cos\theta = \frac{\vec{a} \times \vec{b}}{\|\vec{a}\| \times \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\sqrt{\sum_1^n b_i^2}} \tag{2}$$

where $\vec{a} \times \vec{b} = \sum_1^n a \times b = a_1 \times b_1 + a_2 \times b_2 + \cdots + a_n \times b_n$ is the dot product of two vectors.

As you include more words from the document, it's harder to visualize a higher dimensional space. But you can directly compute the cosine similarity using this math formula.citeinproceedings

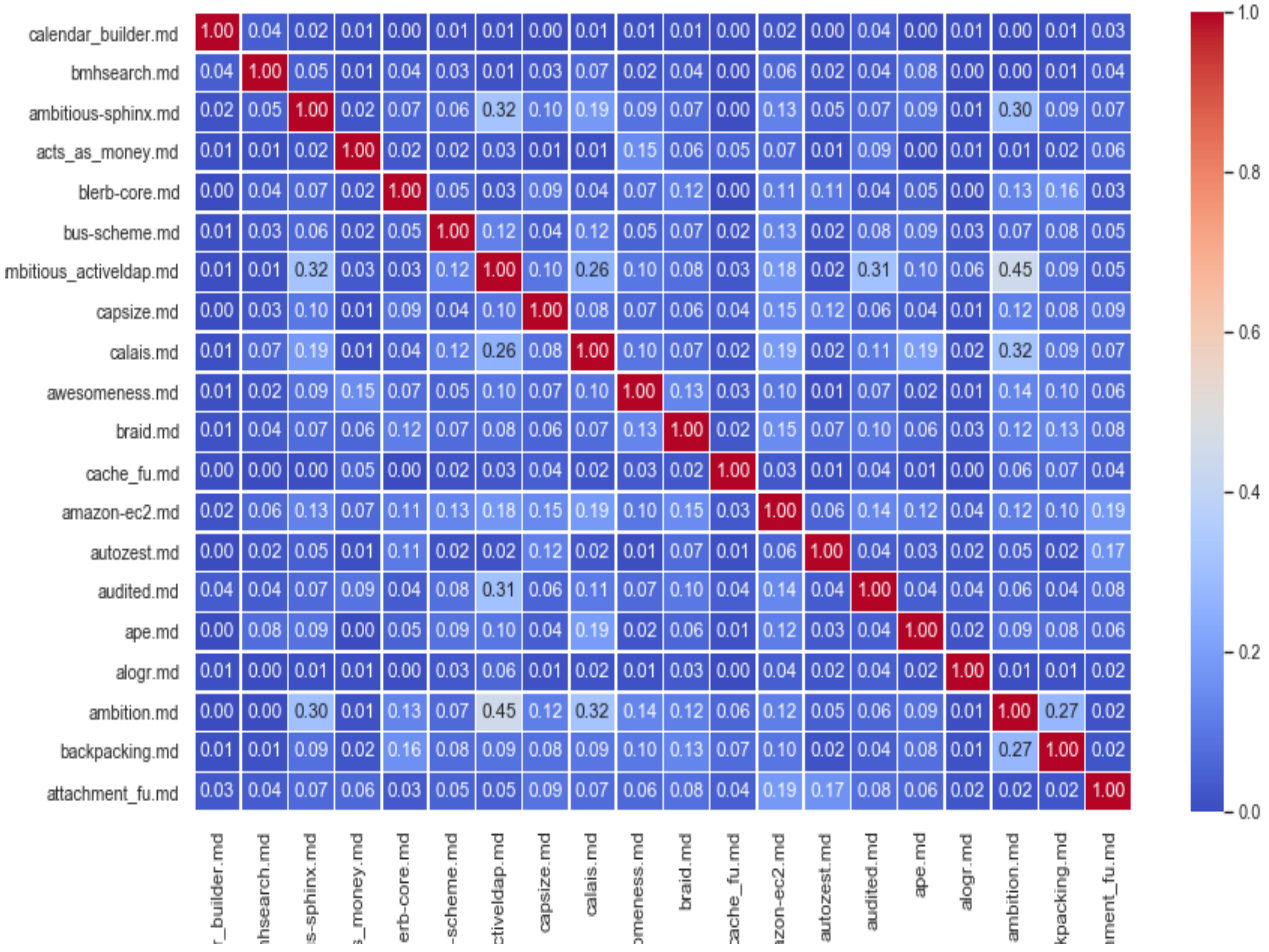

Figure 3: Similarity among 20 Github repertories

In Fig 3 a heatmap table of the similarity among README files of repositories is presented. There are not considerable similarities among these documents. The maximum similarity got was 0.45, which is less then the minimum of 0.6 to consider files likely to be similar.

4 Conclusion

To conclude, in this article, we have performed a study of similarities among public repositories at GitHub to gain an intuitive understanding of the way projects are similar over this social coding platform. A common text processing algorithm TF-IDF (describing document using the frequency of words occurring in them weighted by the importance of the word in the corpus) were used and finally we computed a Cosine Similarity metric to measure the similarity among document. At the end of the project, we found no relevant similarities among these files, meaning that collaborators do not necessarily need to have similar projects, whats turn the study of link prediction between a user and a repository some kind random. In the future, we wish to allocate more resources into extracting and computing statistics for GitHub network of millions of developers contributing to a more extensive set of projects in GitHub.

References

[1] Shahzad Qaiser and Ramsha Ali. Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181, 07 2018.

[2] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic cosine similarity. 10 2012.

[3] Matthew A. Russell. *Mining the Social Web*. 2014.