GH4RE: Repository Recommendation on GitHub for Requirements Elicitation Reuse

Roxana Lisette Quintanilla Portugal[1], Marco Antonio Casanova[1], Tong Li[2], Julio Cesar Sampaio do Prado Leite[1]

[1]Departamento de Informática, PUC-Rio, Rio de Janeiro, Brasil CEP 22451-900
[2]Beijing University of Technology, China
{rportugal, casanova, julio}@inf.puc-rio.br
litong@bjut.edu.cn

Abstract. One of the challenges of requirements engineers is to understand domain issues and elicit requirements effectively. One of the possible strategies is to perform a manual inspection of similar projects to quickly gain leverage of domain concepts underlying the projects. However, this task is time-consuming and limited to the projects at hand. To ensure comprehensive elicitation using more widely available systems, we propose to use GitHub projects as information sources. To handle the large amount of data and facilitate access to suitable sources, we propose the creation of project profiles with useful attributes for requirements engineering, and thereby achieve a meaningful recommendation of projects. In this paper, we describe the GitHub assets to be mined, its implementation and the assessment of our approach by using a corpus of *readmes* related to *Real Estate* projects.

Keywords: Requirements Elicitation; Recommendation Systems; Open Source Repositories; GitHub.

1 Introduction

When dealing with elicitation tasks, one important source of information for requirements engineers is a set of similar projects, because they allow engineers to learn about the domain, their features and contexts in order to be better prepared for more focused requirements elicitation tasks, e.g. interviews/meetings with stakeholders.

GitHub, a repository of open software projects, may help in this scenario. GitHub stores a vast number of projects that describe an application domain through its different *perspectives* (e.g., readmes, issues, comments, source code, and release notes). These *perspectives*, if mined properly, can provide relevant knowledge. However, we cannot rely on manual inspection of projects, since it is time-consuming and unfeasible, given the plethora of projects in GitHub. On the other hand, automated techniques to find similar projects must deal with heterogeneous data in each *perspective*, and this falls into the same problem cited by Castro-Herrera and Cleland-Huang [1], "techniques that work well for recommending carefully categorized books, or movies with ample

X. Franch, J. Ralyté, R. Matulevičius, C. Salinesi, and R. Wieringa (Eds.):
CAiSE 2017 Forum and Doctoral Consortium Papers, pp. 113-120, 2017.

ratings, will not necessarily work well for recommending discussion topics existing in forums or wikis". Therefore, given that we are using the *readme* perspective, techniques to facilitate the finding of meaningful assets in texts must be carefully designed, bearing in mind the requirements engineer needs. This may allow the creation of a meaningful recommendation set of GitHub projects for requirements elicitation purposes.

A common approach in Recommendation Systems is the content-based filtering, which is used to extract features in items (movies, songs, or web pages), and the use of user profiles based on his/her preferences [2]. In this regard, two issues arise. The first issue is that the *readme* is not categorized, as a book or a movie, since it contains free texts with an undefined purpose, i.e. a *readme* may contain information about features as well as installation instructions. The second issue is that users are rating GitHub projects by their end function and not by their *perspectives* separately.

We propose a recommendation approach for GitHub projects by discovering *meaningful* assets on GitHub projects *perspectives*, for which the *readme perspective* will be used as a sample. To supply the lack of user preferences, we propose to: (1) reveal frequent terms that may cause an interaction with the user; and (2) based on his/her choices, cluster and rank projects for recommendation. Our approach relies on a NLP-based approach for Recommendation Systems [3] that specifically filters nouns and proper nouns to reveal important entities that allow for user browsing. As a seed piece, we begin by using a user query "real estate" to retrieve *readmes* from GitHub projects and ordered according GitHub relevance [4]. With this base, the extraction of *meaningful* information is performed.

We believe this approach can support requirements elicitation tasks, as GitHub can be used as a source of information providing projects that acts as domain *viewpoints* of a domain that can be considered before and during software construction [5][6]. Differently from proposals that exploited GitHub during software construction, our approach aims at supporting requirements elicitation, contributing to better projects recommendation in GitHub *perspectives*. We adopt the *real estate* domain as a running example to exemplify our approach.

This paper is organized as follows. Section 2 reviews related work. Section 3 describes the implementation to extract meaningful assets from *readme data*. Section 4 presents the assessment based on the quality of projects recommended to support requirements elicitation tasks. Finally, Section 5 describes the limitations and opportunities of this work.

2 Related Work

Recommendation Systems in Requirements Engineering (RSREs) has been a growing interest in the field of Requirements Engineering (RE) as surveyed by [7]. The authors referenced 23 papers, two of which were highlighted for pointing out an overview of potentials of RSREs. In particular, one stress that Requirements Engineering (RE) usually deals with domain knowledge, which is often vast and evolving [8]. Furthermore, it is mentioned the necessity to process huge amounts of information by stakeholders, which includes what are users' needs, why they are needed, what are competitors offering, what are technological advances and the feasible features [8]. Although not

made explicit by authors in [8], it is important to note that much interesting information about users' needs can be found in documents from similar projects [9], and GitHub provides an opportunity to discover latent related information about user needs, competitors' offerings, as well as technological assets being used.

From the works surveyed in [7], five of them address requirements elicitation [10-14], having that [10-13] as well as [15] are extensions of [10]. To our knowledge, there are a few studies that focus on requirements elicitation using recommendation systems [10-15]. Such works [10-13][15] recommend forums to stakeholders by first eliciting stakeholders' needs, performing mining of *themes* on these texts, clustering, and finally making the recommendations. These works [10-13][15] are similar to ours in the sense that they use an open-source repository (SugarCRM) as a source of information and the *feature requests perspective* to evaluate their approach. The difference with our approach is that they know in advance the existence of *features* or *themes* in the documents they mined, as well as the domain. In this regard, this is in their advantage, as preliminary work by Portugal et al. evidenced the ambiguity issue in natural language free texts, e.g. *real estate* was identified as an analogy in HCI Usability lingo ("the amount of space available on a display for an application to provide output") [9]. Another work mentioned in the survey [7], Lim and Finkelstein [14] use a collaborative-filtering system to identify stakeholders and ask them to indicate other stakeholders to recommend relevant requirements. This collaborative-filtering system was also used in [10-13][15].

Our approach differs from previous work in the sense that we propose a recommendation system using content-based system, which is based on the semantic content of data. On the other hand, as we cannot rely on the user preferences that exist in GitHub, given that projects are mostly ranked for their development purposes, we propose the use of NLP techniques [3] to reveal latent words in *readme* texts that can be candidate keywords for better projects recommendation. Finally, work related to recommendation of GitHub projects can be found in the approach proposed by Guendouz et al [16]. This work is similar to ours in the sense that it predicts useful repositories according to developer needs. Such predictions are done by exploring the *fork* perspective based on users' activity history.

3 Processing GitHub Readmes for Projects Recommendation

In this section, we present the activities that support the recommendation of GitHub projects. These activities, detailed below are implemented in R [17] for our recommendation system, and use a tool (http://corpus-retrieval.herokuapp.com/) for retrieving *readmes* raw data from GitHub given a query [4].

Our approach took as a sample the *readme* perspective of GitHub projects because this *perspective* is the front end to communicate to humans the features a project implemented. However, not all *readme* texts follow this pattern, some of them misses descriptions, and some misses feature explanations and has instructions for installation instead. It is worth noting that some projects even do not have a *readme* text because its creation is not mandatory on GitHub. Despite these shortcomings, as we are dealing

with hundreds of projects, it is possible to filter a large set of *readme* texts that can be used to find meaningful assets as shown in previous works [4][9][18][19].

The **retrieval activity** is used to automatically extract *readme* texts by submitting a query through the GitHub API [4]. The result is a corpus of 2,155 *readme* texts ordered by its default relevance that is given by the GitHub *bestmatch* function. Due to the GitHub's constraints about phrase queries, we applied a *match string* function in R [17][20] and obtained 1,772 *readme* texts. We called it corpus *R*.

The **filter activity** uses the corpus *R* to perform a filtering step that make use of POS-tagging techniques [21][22] to transform unstructured data (corpus *R*) into structured data by distilling important assets, such as verbs, nouns, and proper-nouns. We called it corpus *T*. This activity addresses the usual preprocessing tasks in texts such as removing numbers, whitespaces, and non-alphanumeric terms. Finally, each *readme* text in corpus *T* was exported as a comma-separated values (CSV) file. These files are available at (https://git.io/v9YJ3) for further data exploration.

The **discovering activity** uses the NLP processing approach [3] to discover and reveal frequent words that may attract user attention due to the lack of user preferences for *readme* texts. In this regard, a corpus *NP* was created using the *proper nouns* from corpus *T*. The frequency of words was computed by using *tf-idf* weighting [23], and a Wordcloud visualization technique was applied to display the data [24] (Fig. 1). In corpus *NP*, the word *Zillow* (Zillow, is an online real estate database company) appears to be the most relevant word in the *real estate* domain.

Fig. 1. Frequent proper nouns in *real estate.*

The **preference activity** aims to simulate the lack of user preferences. In this regard, we used the Wordcloud representation (Fig.1) so that the user (a requirement engineer) can choose what is of their particular interest. We have assumed that the user chooses the most relevant word of this domain *Zillow*, and with this, we can filter a subset of *readme* texts. For the *Zillow* word, 61 *readme* texts have at least one occurrence of this word. We called this group corpus *S*.

The **clustering activity** is used to organize a corpus *S* for recommendation. On this subject, we needed to know the optimal number of clusters that was obtained by using the *k-medoids* algorithm [25], then we used the *k-means* algorithm [26] that based on the number of cluster determines the similarity of each group. For corpus *S*, we got two clusters, cluster *A* with one *readme* text and cluster *B* with the rest. Since cluster B has most of the *readme* texts, Cluster *B* is our GH4RE recommendation for the *Zillow* word.

It is important to note that optimal number of clusters depends on the method used to find similarities. The *k-medoids* used in our approach uses the *silhouette* approach [25] which computes clustering for different values of *k* (number of clusters), we set a *k-min=2* and *k-max=15*. For each *k*, calculate the average silhouette of observations (avg.sil). Plot the curve of avg.sil according to the number of clusters k. The location of the maximum is considered as the appropriate number of clusters.

We visualize that our approach allows iterative discovering of latent words in *readme* texts, e.g., once the clusters are identified, each cluster can become the input of the discovery activity to deepen the search of latent words.

4 Preliminary Assessment

In this section, we present an assessment of 20 *readme* texts to verify the quality of our recommendation approach. We queried *"real estate zillow"* on GitHub and selected the first 10 *readmes* (Table 1). We called this group GitHub recommendation. On the other hand, we selected 10 *readmes* from our GH4RE recommendation group. We combined both groups of *readme* texts randomly, and then we asked six users to assess the usefulness of the information in the texts. For such measures, we used the Likert scale technique [27]. The following text was presented to users:

> *Imagine a scenario where a client desires an application, for instance, an application for the Real Estate domain. One of the tasks you may need to perform as a (requirements engineer, developer, project-manager or designer) is the learning about the Real Estate domain. By observing the Wordcloud (Fig. 1), you may perceive that Zillow is an important word in this domain. The excel file (https://git.io/v9YJc) presents 20 links to Readme texts containing information about Zillow. We want to measure the usefulness of information in texts for the concept Zillow and for the scenario described.*

Six people with experience in Software Engineering were selected to perform such assessment. Among these people, three are RE senior researchers, one is an HCI senior researcher, one is a senior developer, and the last one is a project manager. It is important to note that all corpuses used in this work, as well as the assessments files, are available at GitHub for further research and feedback (https://git.io/v9YJ8).

Each readme text is named following the pattern below to keep the traceability to its sources:

Number of original relevance in GitHub.-.userName.-.projectName

From GH4RE recommendation, we ranked the *readme* texts according to the frequency of *Zillow* word in texts, and then we took the top10 *readme* texts (Table 1). On average, each participant took 30 minutes to complete the assessment. Half of the users performed the assessment online. From our recommendation, the first two *readme* texts were rated as *extremely useful* by five people (83% of users). By contrast, the first two *readme* texts of Github's recommendation were evaluated with the lowest scores (*useless* or *not very usefull*). We note that the assesment of 80% of *readmes* from our recommendation range from *somewhat useful* to *extremely useful*, which we consider as a positive result, given the various profile of participants.

Table 1. Top10 Projects Recommended by GitHub and Top10 projects of GH4RE Recommendation

GitHub Recommendation	**GH4RE Recommendation**	**Frequency of *Zillow* in GH4RE**
0001.-.jdemaris.-.real	1380.-.CurleySamuel.-.Thesis	26
0002.-.litianbo.-.AndroidZillowFetch	1357.-.MichaelAHood.-.real_estate_recommender	21
0003.-.annaplusdavid.-.real-estate-comps	0774.-.hanneshapke.-.pyzillow	16
0004.-.hi08060204.-.Real-Estate-Search	1541.-.shawncxc.-.zillow-analysis	8
0005.-.matlai17.-.Zillow-Classification-599	1336.-.verdi327.-.zillow_api	6
0006.-.eternalmothra.-.real_estate_values	1586.-.aminge37.-.prime-group-project	6
0007.-.samidakhani.-.zillow_web_search	2094.-.imFORZA.-.re-pro	6
0008.-.Brian-Koscielniak.-.realtorApp	1073.-.fascinatingfingers.-.ZillowR	5
0009.-.wilk916.-.ZPropertyEvaluator	1349.-.Tim-K-DFW.-.zillow_scraper	5
0010.-.jamesxuhaozhe.-.Real-Estate-Information-Search-Engine-using-Zillow-API-web-based	1534.-.cran.-.ZillowR	5

For the *readme* texts with the lowest score in the GH4RE recommendation, <https://github.com/hanneshapke/pyzillow> and <https://github.com/imFORZA/re-pro>, we manually verified their contents and found that the first one describes a *client package* as well as its functions, and the second is a brief text indicating the features of a tool. As this is information that can be better apreciated by a person with a developer profile, we looked for the assessment of the user with this profile. Contrarily to what we assumed, we found this user rated those *readme* texts as *not very useful*. This result may lead to several intepretations, since this particular assessment (with the developer person) was online, and we could not receive any feedback from him. On the other hand, for the presential assessment performed with a Requirement Engineer, we found that his feedback is suitable for this situation: *"In general I perceived those readmes can be useful in different times, for instance the ones I rated with higher values is because I could easily obtain knowledge about Zillow. However, if my objective after learning is the reuse of source code, for sure I would use the ones I rated lowest, because I know they contained development words"*.

Related to GitHub recommendations, 90% of the *readme* texts were qualified with the lowest scores. This situation supports our belief that GitHub is envisioned for development purposes. We verified the first <https://github.com/jdemaris/real> recommendation and found this *readme* text has few lines and describe the installation of a development package related to the Zillow API. As for the verification of outliers, we found that the unique *readme* rated as *somewhat useful*, <https://github.com/eternalmothra/real_estate_values>, is a brief text explaining one feature related to *Zillow*.

5 Conclusions

GitHub is becoming an ideal information source for research related to Software Engineering. However, most of the works have explored GitHub projects from the viewpoint

of code developers. Despite taking the code developers' viewpoint, the approach proposed by Guendouz et al. [16] is similar to ours in the sense that it predicts useful repositories by exploring the *fork* perspective based on the users' activity history.

Our use of clustering is geared towards recommending potential usefulness of GitHub projects as to empower requirements engineers with domain knowledge that will be useful in performing requirements elicitation. The results so far show that our approach for recommending projects based on the *readme perspective* performs better than the direct querying the GitHub base. Such result is important as it improves our overall goal of using GitHub mining as a key strategy for requirements elicitation.

Future work will continue to improve the clustering strategy and will explore other GitHub *perspectives* from the viewpoint of requirements elicitors. Moreover, we plan to further validate our approach in the context of other domains, involving more participants from both academia and industry. In particular, we would like to encapsulate our approach as APIs for public use, and try to directly get feedback from end users.

Acknowledgement

R. Portugal acknowledges the support of Capes. J.C. Leite and M.A. Casanova acknowledges the support of CNPq. J.C. Leite thanks Faperj (*Cientista do Nosso Estado*) support, as well. Tong Li acknowledges the support of Startup Funding No.007000514116022.

References

1. Castro-Herrera C, Cleland-Huang J.: Utilizing recommender systems to support software requirements elicitation. Proc. 2nd International Workshop on Recommendation Systems for Software Engineering. pp. 6-10. ACM. (2010)
2. Pazzani MJ, Billsus D.: Content-based recommendation systems. In The adaptive web. pp. 325-341. Springer Berlin Heidelberg. (2007).
3. Fleischman M., Hovy E.: Recommendations without user preferences: a natural language processing approach. Proc. 8th Int'l. Conf. on Intelligent user interfaces. pp. 242-244 (2003)
4. Portugal R.L.Q., Roque H.F., Leite J.C.S.P.: A Corpus Builder: Retrieving Raw Data from GitHub for Knowledge Reuse in Requirements Elicitation. 3rd Annual Int'l. Symposium on Information Management and Big Data. (2016)
5. Leite J.C.S.P.: Viewpoints on viewpoints. Joint Proc. of the 2nd Int'l. Software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints' 96) on SIGSOFT'96 workshops. pp. 285-288. ACM. (1996).
6. Leite, J. C. S. P and Freeman P., "Requirements validation through viewpoint resolution. IEEE Transactions on Software Engineering, vol. 17, no. 12, pp. 1253-1269, (1991)
7. Mohebzada JG, Ruhe G, Eberlein A. Systematic mapping of recommendation systems for requirements engineering. Proc. Int'l Conf. on Software and System Process. pp. 200-209. IEEE Press. (2012)
8. Maalej W, Thurimella AK. Towards a research agenda for recommendation systems in requirements engineering. Proc. 2nd Int'l. Workshop on Managing Requirements Knowledge. pp. 32-39. IEEE Computer Society. (2009)

9. Portugal R.L.Q., do Prado Leite J.C., Almentero E. Time-constrained requirements elicitation: reusing GitHub content. In Just-In-Time Requirements Engineering (JITRE). IEEE Workshop. pp. 5-8. IEEE. (2015)
10. Castro-Herrera C, Duan C, Cleland-Huang J, Mobasher B. Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes. Proc.16th IEEE Int'l. Requirements Engineering Conf. pp. 165-168. IEEE. (2008)
11. Castro-Herrera C, Duan C, Cleland-Huang J, Mobasher B. A recommender system for requirements elicitation in large-scale software projects. Proc. Symposium on Applied Computing. pp. 1419-1426. ACM. (2009)
12. Castro-Herrera C, Cleland-Huang J, Mobasher B. Enhancing stakeholder profiles to improve recommendations in online requirements elicitation. In 17th IEEE International Requirements Engineering Conference. pp. 37-46. IEEE. (2009)
13. Castro-Herrera C, Cleland-Huang J. Utilizing recommender systems to support software requirements elicitation. In Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering. pp. 6-10. ACM. (2010)
14. Lim SL, Finkelstein A. StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation. IEEE Trans. on Software Eng. pp. 707-35. (2012)
15. Hariri N, Castro-Herrera C, Cleland-Huang J, Mobasher B. Recommendation systems in requirements discovery. Recommendation Systems in Software Eng. pp. 455-476 (2014)
16. Guendouz, M., Amine, A., & Hamou, R. M. Recommending relevant GitHub repositories: a collaborative-filtering approach. on Networking and Advanced Systems, 34. (2015)
17. Team RC. R: A language and environment for statistical computing. (2013)
18. Portugal R.L.Q, Leite J.C.S.P.: Extracting Requirements Patterns from Software Repositories. In Requirements Patterns (RePa), IEEE 6th International Workshop. (2016)
19. Portugal R.L.Q.: Mineração de Informação em Linguagem Natural para Apoiar a Elicitação de Requisitos. MSc. Dissertation. PUC-Rio University, Rio de Janeiro, Brasil. (2016)
20. Rinker, T. W. qdap: Quantitative Discourse Analysis Package. 2.2.5. University at Buffalo. Buffalo, New York. http://github.com/trinker/qdap. (2013)
21. Schmid H. Probabilistic part-of-speech tagging using decision trees. In New methods in language processing. p. 154. Routledge. (2013)
22. Michalke, M. koRpus: An R Package for Text Analysis (Version 0.06-5). Available from http://reaktanz.de/?c=hacking&s=koRpus (2016)
23. Hiemstra D. A probabilistic justification for using tf×idf term weighting in information retrieval. International Journal on Digital Libraries. Aug 1;3(2):131-9. (2000)
24. Ian Fellows. Wordcloud: Pretty word clouds. Package 2.5. https://CRAN.R-project.org/package=wordcloud. (2014)
25. Kaufman, L. and Rousseeuw, P.J., 1990. Partitioning around medoids (program pam). Finding groups in data: an introduction to cluster analysis, pp.68-125.
26. Jain AK, Dubes RC. Algorithms for clustering data. Prentice-Hall, Inc.; (1988)
27. R.A. Likert. A technique for the measurement of attitudes Archives.