DESIGNING A KNOWLEDGE BASE FOR OSS PROJECT RECOMMENDER SYSTEM: A BIG DATA ANALYTICS APPROACH

Shahabedin Bayati  
*University of Auckland, s.bayati@auckland.ac.nz*

Arvind K. Tripathi  
*University of Auckland, a.tripathi@auckland.ac.nz*

Follow this and additional works at: [http://aisel.aisnet.org/ecis2016_rip](http://aisel.aisnet.org/ecis2016_rip)

Recommended Citation  
[http://aisel.aisnet.org/ecis2016_rip/37](http://aisel.aisnet.org/ecis2016_rip/37)
DESIGNING A KNOWLEDGE BASE FOR OSS PROJECT RECOMMENDER SYSTEM: A BIG DATA ANALYTICS APPROACH

Research in Progress

Shahabedin Bayati, University of Auckland, Auckland, New Zealand, s.bayati@auckland.ac.nz
Arvind K. Tripathi, University of Auckland, Auckland, New Zealand, a.tripathi@auckland.ac.nz

Abstract

Online software engineering repositories like GitHub are great resources of socio-technical data about software development process. GitHub as a large-scale social coding environment contains various types of open source projects. Selecting a suitable project from a developer's perspective is difficult and time-consuming task. In this paper, general Big Data approaches and machine learning techniques are used to analyse GitHub data. Variety of socio-technical metrics and factors are extracted from online repositories for data analysis. We find that data pre-processing plays an important role in the proposed approach for GitHub Mining. Design science research method is applied on the pre-processed data on open source software (OSS) projects to design recommendation system for project selection. Content-Based recommendation techniques are proposed with evaluation mechanism.

Keywords: Big Data, OSS Repositories, Recommender Systems, Project Selection, Design Science
1 Introduction

Open source software (OSS) projects are widely used in various domains. These projects are freely accessible to end-users, software practitioners and developers and are open for their contribution towards development and maintenance. For a developer, selecting a project to collaborate and contribute from the large repository of OSS projects is complex and challenging and depends on various factors. Web 2.0 and social network facilities empower open source developers as a large community in the Web. GitHub as a social coding Web foundation is the largest platform, which hosts a large number of open source projects and software engineering artifacts (source codes, documents, binaries, and etc.). GitHub also facilitates many features for collaboration and social networking for open source software community. As a result of that, the popularity of GitHub in open source community has been growing exponentially.

As the number of projects grows, selecting a project becomes more tedious and complex. Project selection can be examined from different perspectives. We provide following examples to elaborate these scenarios. An end-user wants to use free OSS project, a developer wants to contribute to a project, a software architect wants to select libraries, etc.. In this research we focus on developers view to select a project with Big Data perspective. Mining software repositories (MSR) approaches (Godfrey et al. 2009) can be applied on GitHub repositories for data analysis along with other social network analysis (Thung et al. 2013).

Open Source Project selection is considered in literature (Barkmann et al. 2009; Jarczyk et al. 2014; Smith et al. 2014). Our research is different from other studies in this area based on the scale of dataset, variety of projects, selection goals, selection factors and its socio-technical perspective. In this research we focus on general Big Data approaches to recommend a project from large-scale project set. A limited number of studies used Big Data for MSR purposes and they used costly approaches for a specific problem solving. This study is different in using general Big Data techniques for broader domain of research in large scale MSR. Project recommendation in large-scale is a new contribution of this research which is not studied in literature. Although valuable research on developers motivation for participation are done in information systems (IS)(Subramaniam et al. 2009; Zhang et al. 2013) but selecting among alternative projects is an open area(Hahn et al. 2008). Considering object oriented (OO) software engineering metrics, project historical data in repositories and social collaboration as project selection factors is another contribution of this study; these factors are not used together in literature. Furthermore, another contribution of this study is a flexible modular framework with high performance pre-processing abilities is proposed which consumes GitHub data. This framework is not focused in previous frameworks (Barkmann et al. 2009; Happel et al. 2008). Large scale project technical metric collection and calculation has limited previous research with different perspectives and metrics (Zhang et al. 2014).

This research answers the following questions about project recommendation process 1- How to apply general Big Data approaches on an Internet-scale of open source projects in a form of integrated framework? 2- What kind of social, technical and historical metrics/factors of social coding repositories affect project recommendation? 3- How to implement data pre-processing and cleansing as an important phase in Big Data Analytics in a collaborative coding environment to create a knowledge base for project recommendation? 4- How to use recommendation techniques on extracted knowledge to select a ranked list of projects for a developer? To answer these questions we adopt design science research (DSR) method in IS (Hevner et al. 2004) to design a knowledge base for recommender system through a framework. As a research in progress we used Peffers’s (Peffers et al. 2007) iterative DSR model to propose a Big Data Analytics (BDA) framework and evaluate it through prototyping and implementing a recommender system. Figure 1 shows our study plan based on Peffers’ model.
The process of selecting a project to collaborate is a complex task because of the nature of projects and developers, which depend on various parameters. Selecting a book, movie, music (usual cases for recommender systems) is quite different from a project, because in contrast to consumer products, a project changes over time in terms of required skills and focus. On the other hand, developers’ expertise, skills, interests, and social communications also change over the time as they gain experience. Longer a developer contributes to a project, she can access to higher roles and positions, which depend on programming, communication, and mentorship skills. Since it is challenging to match a developer to a project, a project recommender system is more complex than a simple collaborative filtering recommender system (Allaho et al. 2014; Robillard et al. 2014). There are different types of recommender systems including collaborative filtering (CF), content based (CB), demographic, knowledge-based (KB) and hybrid (Ricci et al. 2011). In this research, we propose a project selection process driven by content of projects and knowledge around it. Although CF is one of the widely used ones in commerce but it relies on static characteristics of products and users rating but for OSS projects, there is no explicit ranking mechanism and developers’ skills and projects’ requirements change over time. We suggest that content based and knowledge based approaches are more applicable for OSS project recommendation.

Since 2008, GitHub has served more than 10 million users and around 24 million Repositories. GitHub involves variety of data formats from various source code types, issues, commits and comments to user profiles. A recent work has defined Big Data with 5-Vs (Volume, Variety, Velocity, Value and Veracity) (Goes 2014) and all of these Big Data characteristics can be found in GitHub. GitHub as a Big Data resource of socio-technical data around software engineering has this ability to be analyzed for extracting valuable information about open source development based on MSR techniques (Xie et al. 2009).

The rest of this paper is structured as following. In the next section a brief literature review is presented. In section 3, we illustrate the proposed Knowledge base framework. Sections 4 and 5 discuss the methodology and implementation approach. Later we present the proof of concepts and preliminary results, followed by proposed project recommendation approach. Finally, conclusion, limitation and future research are discussed.

## 2 Literature Review

We now provide a brief review on related research, which makes the foundation of our work. Relevant literature helps us identify the problems, research questions and context in design science research.
2.1 GitHub Analysis and Mining Software Repositories

GitHub has become popular in research community in last few years. Using data from GitHub, a recent research has examined how developers profile on social media platforms can be used for assessing their reputation and skills by other developers and recruiters (Singer et al. 2013). Another work has investigated the associations and relations of developers’ activities in different software communities. Results showed that developers who made more commits asks less and answer more than others (Vasilescu et al. 2013). A recent study has argued about the role of GitHub in improving group awareness in software engineering (Lanubile et al. 2013).

GitHub is platform that is used for personal storage, web hosting and hosts a large number of inactive, forked or mirrored projects (Kalliamvakou et al. 2014) with low or almost no commits. Extracting real and active OSS projects from this platform is anything but trivial. Using text mining and machine learning techniques, prior research has used a small sample of GitHub projects and analysed the effect of programming language on code quality and bugs (Ray et al. 2014). Prior work has used small number of projects due to challenges associated with mining and analysis of large repositories such as GitHub. A recent work (Thung et al. 2013) has applied Data mining and social network analysis techniques to GitHub data to explore the relationships between developers and projects. Authors have constructed developer-developer and project-project graphs to estimate the strength of relationships that can determine developers’ influence on their projects. However, they only focussed on connection between developers and projects without considering developers’ contributions to the projects. We argue that developers’ activities and communications with projects and other developers are critical for a recommendation system to recommend developers to project. This research aims to develop an artifact to collect and pre-process data that forms the knowledge base of the project recommender system.

2.2 Project Selection

Project selection has been a perennial problem for all the developers. The problem has gotten worse with exponential increase in the number of open source projects. Researchers have proposed project selection process based on project popularity and support (Jarczyk et al. 2014), project content and peer influence (Allaho et al. 2014), developers’ current project portfolio, their skills and expertise and some new areas of interest (Terceiro et al. 2012) or going for first-fit approach instead of best-fit (Hauge et al. 2009). Researchers agree that project selection is a complex problem and is driven by many factors such as developers’ reputation, expertise, project feasibility, etc. among others. OSS project recommendation based on large-scale historical data, using big data technologies and mining software repositories has not been investigated to the best of our knowledge. This research aims to fill this gap.

2.3 Recommender Systems for Software Engineering (RSSE)

Recommender systems are widely used in e-commerce in different contexts using a wide range of approaches (Ricci et al. 2011). For example, recommender systems are used to support software practitioners in development process (Robillard et al. 2014). Recommending an expert developer to resolve a bug based on historical data about other bugs and their resolution process is a general case in RSSE. Most of the applications in RSSE are related to expert recommendation and bug resolution tasks in a limited number of projects (Naguib et al. 2013; Robbes et al. 2013). However, RSSE is also used in some other development tasks and software engineering process like requirement management (Maalej et al. 2009), source code recommendation (Holmes et al. 2005), design patterns suggestion (Palma et al. 2012) and tracking changed files (Zimmermann et al. 2005). Overall, literature lacks a flexible general framework for context-aware and semantically enriched RSSEs for project recommendation using large-scale data (Happel et al. 2008).

2.4 Developers Motivation in OSS

Twenty-Fourth European Conference on Information Systems (ECIS), İstanbul, Turkey, 2016
Open source phenomenon as a successful collaborative community is studied with different perspectives in Information Systems. Understanding users’ motivation in project participation has been a recurring theme. Prior research has argued that intrinsic and extrinsic motivations affect developers’ participation and performance (Roberts et al., 2006) and eventually project success (Crowston et al. 2006; Von Krogh et al. 2006; Subramaniam et al. 2009). Since community plays a significant role in OSS development, community response also impacts user contribution in OSS (Zhang et al. 2013).

With increasing flow of corporate money to OSS communities, recent studies have investigated the role of monetary rewards in OSS development (Atiq and Tripathi, 2014). For example, researchers have found that developers’ extrinsic, intrinsic and community motivations drive their decision to accept monetary rewards (Krishnamurthy et al. 2014). While most of the literature is focussed on developers’ motivations, a few studies have examined how previous participation affects developers’ project selection in OSS (Hahn et al. 2008). Building on prior work on project selection process, we propose a framework for data collection and pre-processing for project recommendations in OSS.

3 Proposed Framework

Figure 2 shows an overview of required components and data flow of the proposed framework. A service oriented application, which remotely accesses online GitHub repository for data collection, is proposed. GitHub data dump accumulate a list of projects (Gousios 2013; Grigorik 2012). Since there are a bunch of repositories, which do not contain real projects, we use these dumps as seeds in our framework, which reduce data processing time. The application uses GitHub RESTful Services and project URL (from data dumps) to retrieve and stores project data in a NoSQL Document DB (MongoDB, Raven DB, CouchDB). Thereafter, data cleaning phase is applied on project data to reduce the data size and make a dataset more approachable. Big Data Analytics approaches are used for distributed data processing on project stored documents. Results are stored in in relational DBMS (MySQL, Postgres SQL, SQL Server, and …). R-DBMS data can be directly analysed with machine learning tools or can export other file formats. As illustrated in previous section, there is a lack of a flexible and scalable analytical framework for OSS project analytics. We overcome this problem by introducing a modular Big Data service oriented framework.

4 Architectural Implementation

In this section we discuss how to implement the proposed framework for OSS project selection on GitHub platform. Following Hevner et al (2004) the architectural implementation of a framework can be evaluated as a proof-of-concept in DSR. Data collection and storage, pre-processing and data mining are the main phases of this implementation. After completion of these phases framework can be integrated. Each module is independent and can be maintained separately. This artifact (framework) is a part of project recommender system for software developers. The validity of this artifact is evaluated through machine learning results and accuracy measure like precision, recall, etc.

Data Collection: We have used GhTorent (Gousios 2013) data dump to obtain the list of the real and active OSS projects. Using GhTorent as a seed, we obtain the original data that includes source code repositories, bugs and feature of the projects, etc. from GitHub using GitHub APIs. GitHub RESTful API returns JSON documents while our GitHub Extractor Application calls the APIs with project URL seeds as inputs. GhTorent helps us to find the real projects because there are large number of non-project repositories in GitHub (Kalliamvakou et al. 2014). MongoDB as a document based database is used to store project’s JSON documents about developers, issues, source code, commits, forks and pulls. MongoDB is used in Big Data research to store
unstructured data. Also it supports Map/Reduce and it can be used on top of Apache Hadoop HDFS distributed file system.

**Data Pre-processing:** Data pre-processing is done in two phases - data cleaning and metric computation. In phase I the stored documents are cleaned in 3 steps regarding to our view in project selection. In first step projects are filtered based on three famous (OO) languages (C++, Java, C#). Although there are more OO languages which are popular in the community like Python and JavaScript, these languages are selected because general software engineering metrics are available for measuring the project quality that use these languages. Further, based on Stackoverflow data, these three are the top OO languages used in the projects created during our data collection period (Bayati et al. 2016). This step can be done in the first stage while retrieving data from GitHub, however, we prefer to do it in this place for three reasons: 1- Inaccurate data about main programming language in GitHub as they changed. 2- A complete dataset of projects can be used in future studies. 3- It seems more approachable, maintainable and classified to push all the pre-processing and cleaning jobs in a component.

In the next step of data cleaning, we first select among all mirrored/forked projects and sort them based on their names, parents, contributors, age, commits, open issues, and watchers. This sorting allows us to remove duplicate projects. Many of the selected projects are not live repositories or do not have any issues for contribution, and so, these repositories are cleaned. Time from last activity and number of contributors are good metrics for this cleaning process.

In metric calculation phase we followed literature on software engineering quality metrics of OO projects source code quality and user motivation in OSS (Jarczyk et al. 2014; Zhang et al. 2014). Table 1 lists all metrics used in this research. All of these historical and technical metrics are useful to generate user and project profiles. As these metrics are independent for each source code file and project, Map/Reduce technique can be applied for computation (Shang et al. 2010). Map/Reduce is a distributed processing technique based on distributed divide and conquer algorithm which is used for large-scale datasets (Shang et al. 2010). It has two main phases "Map" and "Reduce", and also "Shuffling" phase in between. In most cases it works based on a <Key, Value>data format. Code quality metrics are calculated based on Map/Reduce. A sample of Map/Reduce for code complexity calculation for a project is shown in figure 3. To normalize data among projects we followed literature (Jarczyk et al. 2014) to use logarithmic transformation for some of the metrics like project size and also we used function point (Marchewka 2006) to normalize LoC.

---

**Figure 2. Proposed framework**

[Diagram of proposed framework]
After data pre-processing phase, the result of metric computation is stored in a relational database to be analysed by statistical and machine learning tools. R data mining packages and Weka (Hall et al. 2009; Witten et al. 2005) are used for Expectation Maximization and K-Means clustering.

5 Preliminary Results

To evaluate our proposed approach we have implemented a prototype of proposed methodology. We have just focused on a limited number of projects for a period of 2010 to 2011 based on project beginning time. This period is selected because GitHub’s popularity peaked in this period. By selecting projects with time of creation, we can analyse the joining process of new users in future iterations of DSR. Moreover, These projects are likely to become mature by the time of writing this paper. In this study around 21% of total repositories belong to C++ (6%), C# (5.5%) and Java (9.5%). Around 88% of these projects are mirrored/forked projects. After data cleaning, we found a chunk (43%) of dead projects. Clearly these projects should be removed from recommendation list.

We selected 10 main metrics from metrics list based on literature. Then by running Map/Reduce functions on cleaned projects the final outcome stored in R-DBMS table. In the final stage the exported CSV from R-DBMS is used for clustering in Weka and the results are visualized with R. Figure 2 shows the visualized box-plots for the clustered projects. Based on four clusters categorization, which is defined based on elbow criteria (Tibshirani et al. 2001), we have cluster1 4%, cluster2 27%, cluster3 57%, cluster4 12%. The accuracy measures for clustering process resulted in SSE (Sum of Squared Error) 53.7. Cluster0 contains the least number of projects, however, these projects are highly recommended. On the other hand cluster3 projects are inadequate for selection. Also projects that belong to cluster1 can be selected based on the conditions. Table 2 shows the result of clustering evaluation.

<table>
<thead>
<tr>
<th>Clustering</th>
<th>TP rate</th>
<th>FP rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-Means, K=5</td>
<td>0.937</td>
<td>0.023</td>
<td>0.935</td>
<td>0.937</td>
<td>0.936</td>
</tr>
<tr>
<td>K-Means, K=8</td>
<td>0.942</td>
<td>0.018</td>
<td>0.932</td>
<td>0.942</td>
<td>0.936</td>
</tr>
<tr>
<td>K-Means, K=4</td>
<td>0.989</td>
<td>0.003</td>
<td>0.990</td>
<td>0.989</td>
<td>0.989</td>
</tr>
</tbody>
</table>

Table 2. Project Clustering evaluation
6 Project Recommendation

The knowledge base that is created, can be used for recommendation. The clustering results in previous section shows projects that have potential for recommendation (Ricci et al. 2011). The listed projects from appropriate clusters are ready for recommendation process. However, content based approaches can be used for personalized recommendation. User’s current contributing projects features set UserPrjs is a query to recommending engine. Cosine distance similarity measure can find most similar projects to users request (Ricci et al. 2011). Equation 1 represents the cosine similarity function where $A$ and $B$ are projects as vectors and $A_i$ and $B_i$ are their properties (vector elements) which can be selected from table 1. For each user, most recently selected contributing projects represent her interests and most similar projects to that are listed as recommendation. We selected the most recent one as it illustrates the current status of a developer and recommended project status should be more similar to this project. Each project is shown in triple $Prj (H, S, T)$ where $H$ represents Historical factors, $S$ used for Social and $T$ for Technical metrics. To sort and rank the projects with similar result from potential clusters, we used project popularity factor ($\#Stars$).

\[
\text{similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}
\]

As a proof of concept for this iteration of DSR, we implemented and evaluate the results. We selected $Prj[H(\#OpenIssues, Age, P_Lang), T(\#Forks, Size), S(stars)]$ and by using a sample set of users of 300 projects tests the recommendation approach. Precision@10 and Recall@10 which are accuracy metrics for top 10 recommended projects will be used to evaluate this recommendation technique considering users real selection and recommended in comparison with baseline (del Olmo et al. 2008). In Future nDCG (normalized Discounted Cumulative Gain) for analysing the ranking quality and MAE (Mean Absolute Error) also will be used for ranking evaluation.

7 Conclusion, Limitations and Future Works

This research focused on using unsupervised data mining techniques to analyse stored data in large-scale open source repositories. GitHub as an open source repository on Web was selected for data analytics. GitHub is a social coding environment which provides both social and technical features for distributed collaborative open source development. This research proposed a new framework for data collection and pre-processing based on projects’ historical, technical and social data. General Big Data approaches like Map/Reduces and NoSQL are used in data processing phase. The results of analysis are used for project selection/recommendation for
developers. Variaty of related factors are used for clustering projects data and preliminary results of the research are used as a proof of concept of the proposed framework. Based on the cluster analysis the appropriate list of projects for recommendation is created. A content based recommender system used the clustering results to recommend ranked list of projects to users based on the similarity of users’ last selected project and other projects from the clustering results.

Though our initial results are promising, there are some limitations in the current work, which need some investigation in future iterations. In future we are going to use the extracted knowledge to design a hybrid recommender system for project selection process. Also feature selection and finding related features with statistical techniques is another trend. Moreover, adding more longitudinal data about projects and users can enrich our knowledge base. A good research area related to this study is finding the main drivers of project selection process, and examining how developers select projects to make contributions. In addition, other similarity measures can be applied and evaluated along with other recommendation techniques. We have discussed earlier that more evaluation metrics for ranking can be added. For the next iteration of DSR users selected project for different sequential period will be used to evaluate the accuracy of recommendation approach.

References


