Latent Co-Development Analysis based Semantic Search for Large Code Repositories

Rahul Venkataramani
International Institute of Information Technology
Bangalore
rahul.venkataramani@iiitb.org

Allahbaksh Asadullah, Vasudev Bhat, Basavaraju Muddu
Infosys Labs, India
{allahbaksh_asadullah, vasudev_d, mbraju}@infosys.com

***Abstract*—Distributed and collaborative software development has increased the popularity of source code repositories like Github. With the number of projects in such code repositories exceeding millions, it is important to identify the domains to which the projects belong. A domain is a concept or a hierarchy of concepts used to categorize a project. We have proposed a model to cluster projects in a code repository by mining the latent co-development network. These identified clusters are mapped to domains with the help of a taxonomy which we constructed using the metadata from an online Question and Answer (Q&A) website. To demonstrate the validity of the model, we built a prototype for semantic search on source code repositories. In this paper, we outline the proposed model and present the early results.**

***Index Terms*—Software repository analysis and mining, source code repositories, semantic search, Human aspects of software evolution**

I. Introduction

Distributed software development and the resulting need for collaboration among developers have made software code repositories with version control ubiquitous in the software development process. This has resulted in millions of projects stored in popular source code repositories like Github[1].

For easy retrieval of projects in such large repositories, it is necessary for semantics like the *domain* of a project to be captured. However, conventional source code repositories do not capture such semantics. For example, the query "distributed systems" is interpreted by the search engine of code repositories as a text string rather than as a *domain*. We define *domain* as a collective notion that exists among a statistically significant population of the developer community about a group of projects belonging to a shared concept. The domain is not an implementation technology but a concept to which a number of projects and technologies belong. A technical professional with relevant experience would immediately note "Hadoop", "Voldemort", "STORM" among other projects as one of the top projects belonging to the domain "distributed systems". Developers are aware of the *domain* of the projects to which they contribute. However, they might not explicitly mention the *domain* in the *wiki* page of the project due to a number of reasons ranging from carelessness to the domain being obvious given the context. This semantic information about the *domain* of a project is lost when the project is stored in a code repository. This makes retrieval of projects from a source code repository on the basis of *domain* a challenging problem.

A model to identify the domains to which projects belong is essential in the development of a semantic search for code repositories. A few systems like SourceForge[2] have partially overcome this problem by forcing the users to annotate the projects with tags before uploading them. However, since the categories are predefined by humans, updation and maintenance of such categories requires more effort. Classification based on similiarity measures calculated from source code artifacts [9] cannot be always used because of restricted access to source code in many corporate repositories. In such approaches, every programming language requires additional engineering effort and developing tools for every programming language is practically infeasible.

In this paper, we propose a model to capture the notion of *domain* of projects belonging to a source code repository by mining the network formed between developers contributing to same project. We call this network as latent co-development network. We constructed a taxonomy from a folksonomy dataset which was used in the development of semantic search engine. The model that we propose can also be used for other allied applications including categorization of projects, measuring popularity of projects among the developer community, skilled recruitment etc.

In section II we present a brief overview of related work. Section III describes a model for identifying domains of projects. Section IV describes the architecture of the semantic search that we built. Section V discusses the experimental setup and the initial results obtained. Section VI outlines future work and some conclusions.

II. Related Work

A number of attempts [4], [5], [7] have been made to extract topics from projects through the use of source code artifacts. Latent Dirichlet Analysis on source code artifacts has been popularly used in the software engineering community to extract topics from projects. However, in the current work we hold the view that mining latent co-development network reveals sufficient semantics to understand the domain

[1]http://github.com

[2]http://sourceforge.net

of projects. Mining the developer contribution network from source code repository has been used to solve a wide variety of problems ranging from understanding collaboration and influence dynamics [3], predicting software failures [8] etc. We are not aware of any other model to identify domains in a code repository using developer project contribution network.

III. The model

Given a source code repository consisting of many projects the problem we address is the following: *find the domain(s) to which the projects belong.*

The proposed model can be described briefly in the following sections:

- Cluster detection of similar projects in code repository
- Mapping project clusters to a *domain* taxonomy

A. Cluster Detection of Similar Projects in Code Repository

In a source code repository, a number of projects belong to a domain. Groups of such related projects are detected and then assigned the respective domains to which they belong. We analyze the latent co-development network in code repository to detect clusters of similar projects. Bird et al. [1] claim that the software components within a project that are developed by the same developer were implicitly related. We extend this claim to software projects in a source code repository and argue that the number of common developers between the projects represent the degree of similarity between them. Formally, *If projects P_1 and P_2 share 'd' common developers and $d > k$, where k is a minimum threshold then the two projects are implictly related to each other.*

We further claim that this similarity is because of the same or related domains that the projects belongs to and not only because of the implementation technology used. Empirically, we observed that such behavior among the developers is more pronounced in popular domains like distributed systems, NOSQL databases, machine learning etc.

An affiliation network(see Fig.1) between developers and projects captures the latent co-development relations in a source code repository. An affiliation network is an undirected bi-partate graph with two types of nodes - developers D and projects P and can be formally defined as $A = (D, P, T)$. D and P are the two sets of vertices that represent developers and projects, respectively. T is the set of edges that represent the contribution of developers towards projects $T \subseteq \{(d,p) \mid d \in D \& p \in P\}$. Using this representation of a source code repository, we construct a project similarity graph $G = (V, E, w)$ to capture the similarity between projects in a source code repository(see Fig.2). V is the set of vertices that represents all the projects in a code repository. E is the set of edges representing similarity between projects. The function $w : E \rightarrow \mathbb{N}$ represents the degree of similarity. The degree of similarity between projects is represented by a count of common developers. The weight of a edge E_{ij} between two projects P_i and P_j is calulated by $w_{ij} = \{|d \in D| \mid (d, P_i) \in T \& (d, P_j) \in T\}$.

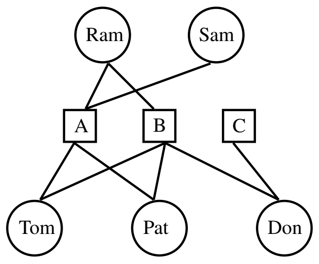

Fig. 1. An example of developer contribution affiliation network. Circles denote developers and rectangles denote projects. An edge indicates the contribution of a developer to a project.

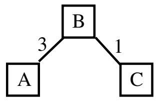

Fig. 2. Project similarity graph constructed from the the developer contribution network in Fig. 1. The edge weights indicate the number of common developers between the two projects.

Clusters of projects $C = \{C_1, C_2, ..., C_n\}$ are extracted from the project similarity graph G using the community detection algorithm proposed by Vincent et al. [2].

B. Mapping project clusters to a domain taxonomy

Each of the project clusters obtained have to be mapped to domain(s). The *wiki* page describes the goals of the projects and its implementation technology. Since there is no explicit mention of domain names in the *wiki* pages, there exists a need for a taxonomy to encompass all the terms in software engineering with the domain names subsuming the respective technologies used for implementation. For example, *Haskell*, *Scala*, *Lisp* belong to the domain of *functional programming languages*. A taxonomy is a formal representation of a set of concepts and the relationships between the concepts.
We adopt the following definiton proposed by Kaipeng et al. [6]

DEFINITION 1 (Taxonomy). *A taxonomy is a structure $\mathbb{O} = (I, root, \preceq_I)$ consisting of i) a set I of concept identifiers, ii) a designated root element, i.e. root representing the top element of the iii) upper semi-lattice $(I \bigcup \{root\}, \preceq_I)$ called concept hierarchy.*

The nodes in the higher levels of hierarchy represent the domain of the project cluster while the lower levels are occupied by the implementation technologies and tools. This taxonomy can be generated by a number of means like manual curation, inducing a taxonomy from a folksonomy dataset etc. Assuming the presence of a taxonomy to satisfy our requirements, we discuss the mapping of a cluster of projects to concept(s) in a taxonomy. The mapping between project clusters and concepts in the taxonomy is given by the relation $M : C \times I \rightarrow n$. n denotes the strength of mapping between a cluster and a concept in the taxonomy. The project descriptions of each of the projects in the cluster are consolidated and this singular description of a cluster is mapped to concept identifier(s) in the taxonomy. This mapping from a cluster identifier to concept identifier(s) in a taxonomy can be performed in a number of ways depending upon the application for which the model is used for.

IV. Application of the Model : Semantic Search

The proposed model can be used for a number of applications like categorizing projects in a repository, project recommedations, skilled recruitment etc. In this section, we demonstrate the application of the model to implement semantic search over large code repositories to browse through the projects belonging to a specified domain in a code repository. In the following sections, we describe the architecture of the semantic search engine that we built using our proposed model.

A. Mapping clusters of projects to a taxonomy

We constructed a taxonomy for software engineering by converting the folksonomy dataset obtained from a Q&A website by using the technique outlined by Liu et al [6]. Most of the programmers contributing to source code repositories also posted questions and answered queries on such technical forums. Hence, the list of tags extracted from such a dataset is fairly exhaustive.

From a partial dump of Github, we constructed the project similarity graph G. We use the community detection algorithm proposed by Vincent et al. [2] to obtain clusters of related projects. We used this algorithm because the quality of communities detected are good in terms of modularity. The *wiki* pages of projects in the cluster are aggregated to form a single description of the cluster. From this aggregated description of a cluster of projects, we restricted our vocabulary of *technical* terms to the tags used to annotate questions in a the Q&A website. Depending on the frequency of the *technical* terms in a cluster descriptor, each project cluster is assigned to concept identifier(s) in a taxonomy.

B. Query Expansion

A user enters a query which represents a $domain$. A query is input to the query expansion engine which expands the given query to an extended set of semantically related terms. This helps the search engine to augment the query with a set of semantically related words to facilitate better searching. We use the metadata from a large Q&A website to find semantically related terms to the given query. A Q&A website is a dataset of the form $F : (Q, T, \alpha)$, where Q represents the set of questions, T the set of tags used to annotate the questions and the binary relation between the tags and questions is represented by $\alpha \subset Q \times T$. Using this dataset, we constructed a tag co-occurence graph $G = (T, E, w)$ to model the developer community's perception of related tags. Edges E represent the co-occurence between the tags T and w represents the degree of co-occurence between tags. Each term in the query S is mapped to a tag T_i in the graph F. All the neighboring tags of the T_i are ordered according to the

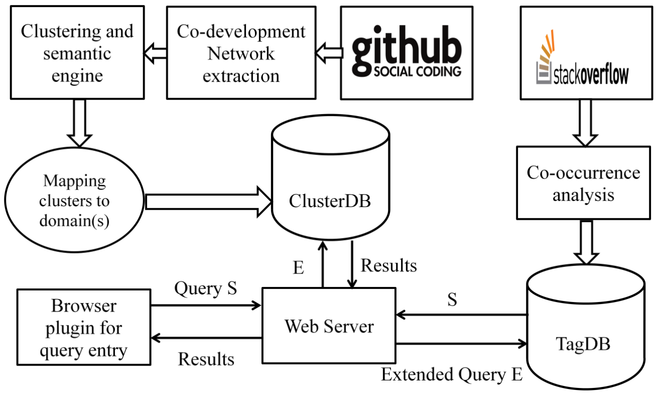
Clustering and semantic engine
Co-development Network extraction
github SOCIAL CODING
stackoverflow
Mapping clusters to domain(s)
ClusterDB
Co-occurrence analysis
E
Results
Query S
S
Browser plugin for query entry
Web Server
TagDB
Results
Extended Query E

Fig. 3. Block diagram of the system architecture

edge weights with the tag T_i . We use the top k neighboring tags to augment the query. However, since all the terms in the neighborhood are not equally important they are weighted by a factor w_{ij}, representing the weight of the edge between the given query tag T_i and the neighboring tag T_j. We have listed programming languages as stopwords since they are ubiquitous in a dataset of this nature. If the query term contains more than a single word representing two or more tags in the graph G, the intersection of neighbors of the terms are given higher precedence by weighting them appropriately.

C. Ranking clusters of projects

We use a term frequency-inverse document frequency(TF-IDF) measure to denote the relevance of a cluster identifier to a project cluster. The score s of a cluster C_i for a query S is defined as the sum of the product of the weights of each term in query with the matching domain term. $s(C_i) = \sum_{s \in S} w(s) \times M(s)$. The clusters are ordered according to the obtained scores. The projects within a given cluster are displayed according to a parameter which can be changed at the user's discretion. Some of the parameters include best string match, last updated project, most number of forks, most number of watchers etc.

V. Datasets and experimental setup

In this section, we first describe the dataset and the experimental setup used in the experiments, and then present the evaluation results, in brief.

We used two datsets for conducting experiments: StackOverflow dataset for taxonomy construction and a partial repository of Github to test our hypothesis. StackOverflow is a collaboratively edited Q&A website for programmers and is one of the most used sites for posting technical questions. We used the latest data dump for StackOverflow at the time of writing the paper which was updated till 1st March,2013. The raw dataset consists of 4,189,241 questions and 32,051 unique tags to annotate the questions.

We used the Github REST API to retrieve a partial list of projects to form our experimental source code repository. Our partial dataset consists of 30,906 Java projects contributed to

TABLE I
EXPERIMENTAL RESULTS FOR SIMILARITY VALUES

Similarity measure	Similarity
Jaccard	0.72
Overlap	0.85

by 22,322 developers. Using this dataset, we construct the project similarity graph G. We obtained 150 project clusters after applying the community detection algorithm. We ignored the clusters with less than 5 projects because such clusters are typically formed because of contributions from a single developer and is mostly written for personal or academic purposes. The setup for the query expansion engine and ranking of clusters is hosted on a internal server. The results are displayed to the user through a Google Chrome extension which modifies the default Github search engine to use our implementation instead.

A. Evaluation results

The evaluation of our results is carried out in two stages. In the first stage, we performed a user evaluation to validate the relevance of domain(s) detected for each project cluster using our proposed model. We requested developers to tag each cluster with names of domain(s) that they think the cluster belongs to. In order to avoid the bias of a single developer, we used crowdsourcing techniques to tag the clusters. We selected 15 developers working across various teams at Infosys to tag 10 clusters of projects each in the first round. In the next round, the clusters were exchanged between the developers without their knowledge and they were requested to perform the same exercise again. The tags obtained from the two rounds were consolidated to represent the domain(s) which the volunteers percieved. We evaluated the similarity of the consolidated set of tags obtained from developers for a cluster with the domains obtained from our model using the Jaccard and overlap similarity measures. Let X_1 represent the domains obtained from our model and X_2 represent the consolidated set of tags for a cluster obtained from developer tagging. The Jaccard similarity is given by:

$$J(x_1, x_2) = \frac{|X_1 \cap X_2|}{|X_1 \cup X_2|} \quad (1)$$

The overlap similarity is defined according to the relation:

$$\sigma(x_1, x_2) = \frac{|X_1 \cap X_2|}{min(|X_1|, |X_2|)} \quad (2)$$

The average value for the similarity measures for all the project clusters is listed in Table I. The high values of similarity measures obtained indicates a strong correlation between the domain(s) obtained by our model and the existence of a notion of domain(s) among the developer community.

In the second stage of evaluation, we observed the results obtained from semantic search qualitatively. We tested our semantic search engine with a few queries, the results of which are displayed in Table II. Given the limited source repository for testing, it is encouraging to find that most of the projects are closely related to the query domain. We manually compared the search results with Github's native search engine and observe that our results are semantically more relevant to the given query.

TABLE II
SEARCH RESULTS FOR A SAMPLE OF QUERIES

Query	Search Results
social networks	Twitter4J, FourSquareJavaAdaptor, Trie4j Face4j, openGraphJava, prettyTimeSocial
knowledge representation	libSBOLJava, libSBOLRDF, javaOWLSensor JavaOWLSolver, RESTOWLCommon
games	pinBall, Sudoku, MineSweeper Tetris, Pentago, TicTacToe
cloud computing	java-mongoDB-connector, javaAmazonDB jClouds, javaCloudFoundry, heroku, rabbitMQ

VI. Conclusion and Future Work

Through this work, we demonstrated a technique to mine latent semantics in a source code repository using the developer contribution network. We formally defined the notion of a *domain* in software code repositories and proposed a model to detect such domains. We corroborated our hypothesis with experimental evidence. This work is a part of a larger vision to model the entire software engineering ecosystem comprising of source code repositories, developers, question and answer sites, technical blog posts, wikis, and use the information to arrive at improved semantics. In the future, we plan to explore other semantics that can be extracted from source code repositories and demonstrate the same through applications.

Acknowledgment

The authors would like to thank Kshitiz Bakshi for his contribution in developing the browser plugin. We would like to express our gratitude to Dr. Srinivas Padmanabhuni for his guidance throughout the course of research.

References

[1] C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu. Putting it all together: Using socio-technical networks to predict failures. In *ISSRE*, pages 109–119. IEEE, 2009.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *JSTA*, 2008(10):P10008, 2008.

[3] B. Heller, E. Marschner, E. Rosenfeld, and J. Heer. Visualizing collaboration and influence in the open-source software community. In *Proceedings of the 8th WRCE*, pages 223–226. ACM, 2011.

[4] E. Linstead, S. Bajracharya, T. Ngo, P. Rigor, C. Lopes, and P. Baldi. Sourcerer: mining and searching internet-scale software repositories. *Data Mining and Knowledge Discovery*, 18(2):300–336, 2009.

[5] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi. Mining concepts from code with probabilistic topic models. ASE '07, pages 461–464, New York, NY, USA, 2007. ACM.

[6] K. Liu, B. Fang, and W. Zhang. Ontology emergence from folksonomies. In *CIKM*, pages 1109–1118. ACM, 2010.

[7] G. Maskeri, S. Sarkar, and K. Heafield. Mining business topics in source code using latent dirichlet allocation. In *Proceedings of the 1st India software engineering conference*, pages 113–120. ACM, 2008.

[8] M. Pinzger, N. Nagappan, and B. Murphy. Can developer-module networks predict failures? In *Proceedings of the 16th ACM SIGSOFT FSE*, pages 2–12. ACM, 2008.

[9] T. Wang, G. Yin, X. Li, and H. Wang. Labeled topic detection of open source software from mining mass textual project profiles. In *LAMDA*, pages 17–24. ACM, 2012.