Boa Meets Python: A Boa Dataset of Data Science Software in Python Language

Sumon Biswas, Md Johirul Islam, Yijia Huang, and Hridesh Rajan
Department of Computer Science
Iowa State University
226 Atanasoff Hall
Ames, IA, USA
{sumon, mislam, hyj, hridesh}@iastate.edu

***Abstract*—The popularity of *Python* programming language has surged in recent years due to its increasing usage in *Data Science*. The availability of *Python* repositories in *Github* presents an opportunity for mining software repository research, e.g., suggesting the best practices in developing *Data Science* applications, identifying bug-patterns, recommending code enhancements etc. To enable this research, we have created a new dataset that includes 1558 mature *Github* projects that are developing *Python* software for *Data Science* tasks. By analyzing the metadata and code, we have included the projects in our dataset which use a diverse set of machine learning libraries and managed by a variety of users and organizations. The dataset is made publicly available through *Boa* infrastructure both as a collection of raw projects as well as in a processed form that could be used for performing large scale analysis using *Boa* language. We also present an initial application to demonstrate the potential of the dataset that could be leveraged by the community.**

***Index Terms*—MSR, Boa, AST, machine learning, data science, open source repositories, program analysis**

I. Introduction

Machine learning tools and techniques are becoming gradually prevalent in data-intensive software projects. Among various languages used for Data Science, *Python* has become a popular language because of its large collection of libraries to analyze and organize data. Mining such *Python* projects would be helpful to improve language design, library enhancements, bug detection as well as open new research directions. For example, by analyzing programs from thousands of data science projects, we can suggest the best library for performing specific tasks, find recurrent bugs, improve certain APIs, etc.

Inspired by this need, we have created a dataset from *Github* data-science repositories that are using the *Python* language. We have stored the AST of parsed *Python* programs from each revision along with the metadata into the dataset.

Collecting appropriate projects from *Github* repositories and analyzing them has been an effective method for mining software repository research [1]. DaCapo benchmark [2], the New Zealand Digital Library project [3], Software-artifact Infrastructure Repository (SIR) [4], are good examples of datasets that contain C, C++, C# and Java projects. These datasets do not include the project's commit history. There are also some datasets that are not open-source. Another category of datasets include the project's commit history. *Boa* [5], GHTorrent [6] and Java Qualitas Corpus (JQC) [7] are some examples of available curated datasets with history. *Boa* and JQC both provide additional information about the projects in the form of metadata. *Boa* also provides its own domain specific language to write a query on all the revisions of *Github* projects. While most of the datasets have focused on C, C++ and Java, few datasets also include *Python* projects such as GHTorrent [6], Open Hub [8], work of Orru *et. al.* [9]. However, to our knowledge, no dataset is publicly available that contains curated *Data Science* projects written in *Python*.

We created this dataset for *Boa*, a domain specific language and infrastructure for MSR [5]. *Boa* provides a platform for writing program analysis queries and abstracts away details of software repository mining. One can use the web interface of *Boa* to write queries, select the dataset and submit a job to Hadoop cluster. The user does not worry about data collection, storage, and concurrency concerns that are handled automatically by the platform. A *Boa* job can be shared and others can reproduce the result. *Boa* has its own types (Project, CodeRepository, and Revision), built-in methods (getast, getsnapshot, etc.) and other quantifiers (foreach, exists) to write queries on the dataset. A program is parsed and translated into a custom representation including types such as Namespace, Declaration, Method, Variable, Statement, and Expression [10]. A *Boa* program is written to run a query on the dataset. The *Boa* program is converted into a Hadoop [11] MapReduce [12] program and submitted to the Hadoop cluster to run in parallel. The details of distributed execution are also abstracted away from the user.

The main goal for generating this dataset was to enable MSR research on data-science programs. For dataset generation, using the GitHub REST APIs, first, we have filtered the projects that have *Python* as the main language. Then we have filtered the projects that perform data science tasks. For this filtering we have applied two methods: 1) search for data science related keywords in the description of the project, 2) filter projects that use machine learning and data science related libraries. To include only mature projects in our dataset, we also filter projects that have at least 80 stars. By following these filtering criteria, we ensure that the dataset incorporates high-quality top 1558 data science repositories from GitHub written in *Python* language. We also extend the data schema used by the *Boa* infrastructure, originally designed for Java-

like languages, to support the *Python* language and have written data conversion tools for converting raw data into the Hadoop sequence file format used by *Boa* to facilitate scalable queries.

To our knowledge, this is the first dataset that includes data-science projects written in *Python*. Machine learning packages and APIs are changing very rapidly to introduce new features and enhance the existing ones. *Python* is also a well-maintained language that is evolving quickly. Our dataset will help researchers study the state-of-software-engineering-practice followed in open source data science projects.

The key contributions of the paper are:

1) a large dataset for analyzing *Python* data science projects that contains 1558 *Github* open-source projects and 4,977,680 revisions of *Python* files.
2) A data schema for efficiently storing this data in Hadoop sequence file in order to make it memory efficient and parallelly accessible.
3) Dataset is publicly available on *Boa* web-based infrastructure [5]. One can write MSR queries using the *Boa* language and submit the job to Hadoop cluster for further processing.

II. Methodology

A. Data Source

We have collected our dataset from *Github*. Source code in *Github* is organized into repositories, branches and commits which allows retrieving every revision of a program. Apart from the source code, *Github* also provides meta-data about the project such as developer information, commit date, commit log etc. that help to answer a large set of research questions. We filter repositories as per our interest to create the dataset suitable for further research on MSR for Data Science software.

B. Data Collection and Preprocessing

Github provides several REST APIs to search and collect meta-data and source code of the repositories. An overview of the repository selection and filtering procedure is shown in Figure 1. Using the APIs, first we collect all the repositories having *Python* as the main language and having more than one star. To bookmark and follow new updates of a project, *Github* users 'star' a project. Following several prior works in this area, we have also used the number of stars as the indication of matured and popular project. So, we used this metric to further filter good quality projects. For each project, we collect the meta-data and other API URLs for the repository in the form of a JSON file. By following this method, we have collected JSON files for 343,607 *Github* projects. These projects are original (not forked from another project), use *Python* as the primary language and have more than one star.

The next step is to identify mature data-science projects. To filter the data science projects we have followed two methods. First, we filter the projects having data science related keywords such as machine learning, deep learning, neural network, image processing, artificial intelligence, etc. in the

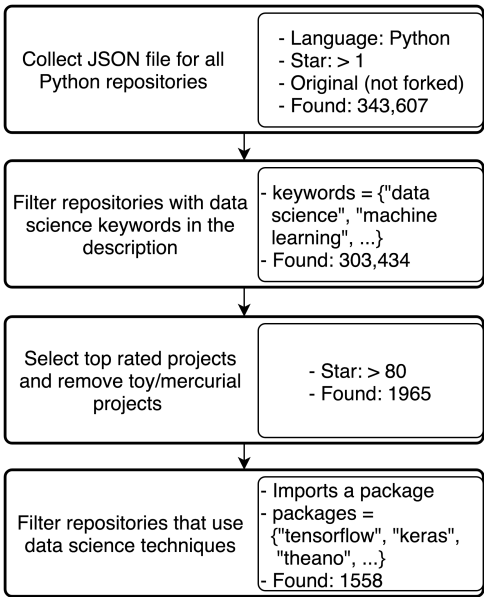

Fig. 1. Repository selection procedure and criteria

Github description of the repository. A keyword search with a wide list of keywords in the description has the possibility to be imprecise. Therefore, we undertake a second strategy. We filter the projects that use machine learning libraries such as: Tensorflow, Keras, Theano etc. To include high-quality projects in the dataset, we have removed projects having less than 80 stars. Note that the number of stars has been chosen to keep the dataset large enough to cover matured projects that use popular data science libraries. Finally, we filtered JSON files for 1558 projects. These JSON files are the input to our dataset generation.

C. Data Generation

For each chosen *Github* repository, the data generation starts with caching all the meta-data and cloning the repository locally. The *Python* parser, implemented in Java, is used to parse the source code retrieved by *JGit* from the local repository into AST data. Then, we utilized data writers generated by the Protocol Buffers compiler to convert the AST data into *SequenceFile*s. Protocol Buffers are an extensible mechanism developed by Google that was designed to serializing structured data fast compared to other formats like XML. The *SequenceFile* stores data in a special format similar to map which stores key-value pairs. In this case, the key is the project and the value is the binary representation of Protocol Buffer message. Finally, the generated *SequenceFile* is transferred to Hadoop Cluster and is made accessible to *Boa* queries.

D. Mapping Python AST to Boa AST

We read each revision of a *Python* file and parse with the appropriate parser. While parsing, we identify the *Python* version used in the program and attach a version tag with each file. Two different parsers have been used to parse *Python* 2 and *Python* 3 files. The next step is to convert *Python* AST to

predefined *Boa* AST format. *Boa* has designed its AST format in a flexible way so that it can incorporate most of the AST nodes from different languages. However, *Python* has different syntactical elements and structure such as *lambda* statement, *with* statement. As a result, we had to add a few new AST nodes and change few existing structure of *Boa* schema. For example, in *Python* a function can be defined inside a function. To achieve that we changed the *Method* node in *Boa* so that it can hold *Method*s recursively.

The top-level node which holds a program file is named ASTRoot in *Boa*. The child nodes under ASTRoot are listed in Table I. The Namespace node holds the qualitative path to the file and other Declarations such as import statements, classes. A declaration of Class kind holds the *Python* functions as Methods which in turn holds other statements and expressions.

E. Data Storage

The storage strategy is the same as other dataset storage methods of *Boa*. The *SequenceFile* data is populated into a distributed database called HBase tables [11]. HBase is provided by Hadoop which is an open source implementation of Google's Bigtable [13]. From these tables another data structure *MapFile* is generated which generates an index file. These generated files and HBase tables are the input to the dataset mining queries.

III. Data Description

A. Metrics

The dataset contains 1558 repositories developed by 9839 individuals users. Figure 2 presents the important metrics of the dataset. The dataset also contains projects that use at least 33 data science libraries including Pytroch, Caffe, Keras, Tensorflow, XGBoost, NLTK, StatsModels, Matplotlib etc.

Metric			**Count**
All repositories	Owner	Individual user	1208
		Organization	350
	Total		**1558**
Revisions			557,311
Python Files (all revisions)			4,977,680
Python Files (latest snapshot only)			86321
Developers			9839

Fig. 2. Metrics of the dataset

B. Data Schema

The dataset is created under the pre-defined data format of Boa. The fields in the dataset for storing meta-data about the repository are shown in Table I.

For storing the parsed AST from the source code, the dataset has following fields listed in Table II. The fields with the *kind* attribute are the union of different record structure. For example, the *kind* of a *Statement* is an enumerated type and based on the value of the *kind* (e.g., IF, FOR, BREAK), other attributes are added to the Statement.

TABLE I
Domain-specific types for storing repository, based on [5].

Fields	**Attributes**
Project	id, name, created_date, code_repositories, ...
Repository	url, kind, revisions
Revision	id, log, committer, commit_date, files
Person	username, real_name, email
File	name, kind

TABLE II
Domain-specific types for source code, based on [5].

Fields	**Attributes**
ASTRoot	imports, namespaces
Namespace	name, modifiers, declarations
Declaration	name, kind, modifiers, parents, fields, methods, ...
Type	name, kind
Method	name, modifiers, return_type, statements, ...
Variable	name, modifiers, initializer, variable_type
Statement	kind, condition, expression, statements, ...
Expression	kind, literal, method, is_postfix, ...
Modifier	kind, visibility, other, ...

IV. Usage

A. Boa Web-based Interface

The dataset can be accessed through *Boa* website [14] to write queries and submit them to *Boa* cluster for execution. An overview of this web-based interface is presented in Fig. 3. There are three steps to executing boa queries on the *Python* dataset. First, log on to the *Boa* website as a registered user. Then navigate to user menu and choose *Run Examples* from the left panel. Second, write a query under the *Boa Source Code*. If researchers are not familiar with the language, the example boa programs can be utilized by clicking the *Select Examples*. Third, select 2019 February/Python dataset in the drop-down list under *Input Dataset* and run the query.

A client API is also provided to programmatically access the *Boa* infrastructure. This allows researchers to use different languages to send a *Boa* query, like SQL queries embedded in other languages, and retrieve results back for analysis.

B. Boa Query and Output

The *Boa* program presented in Fig. 3 is for counting the most used libraries in the dataset. The *Boa* program contains an input project *p* and an output *topimport* as a top aggregator (line 2). A depth-first search (DFS) traversal strategy is implemented in the *visit* function (line 5) with an input project *p* and a specified *visitor*. Before traversing the whole *AST* under *CodeRepository* node, the visitor only looks at the latest snapshot (line 8). For each file in the snapshot, the program uses the same visit strategy (line 10) to aggregate different libraries in *Namespace* nodes (lines 13-16). The top aggregator selects the 10 highest weighted results as output.

The submitted *Boa* jobs can be accessed in the left panel under *Job List*. The corresponding job status such as job ID, created time, source code, compile/execution log is provided by choosing a specific job. If the job is finished without any

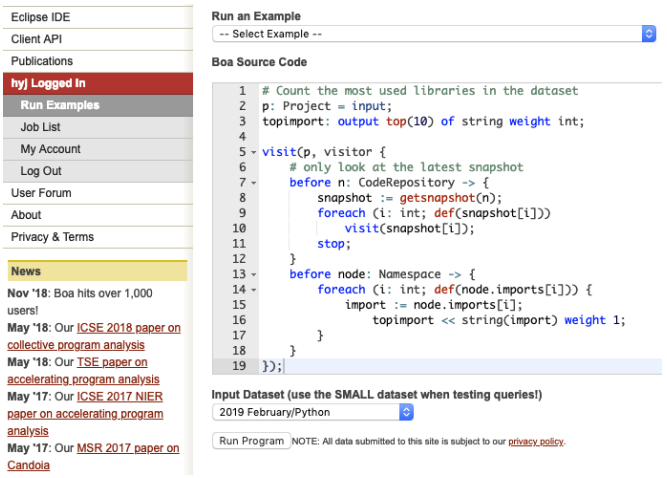

Fig. 3. Boa Web-based Interface

compilation/execution error, the result can be found under the tab *View Job Output*. Researchers can also download the output as a text file for further analysis.

C. Using the dataset without the web interface

The dataset can be accessed outside of the *Boa* infrastructure as well. Our dataset is in the format of Hadoop sequence file that can be read or written using protocol buffer reader/writers. If parallel processing over this dataset is desired, then one would need to use threads or write *MapReduce* [15] tasks from scratch to analyze the data as shown in [5]. Another method to use the dataset outside *Boa* infrastructure is to get the publicly available *Boa* compiler and run queries on the dataset directly after building the compiler in the target environment.

V. Applications

To show the potentials of the dataset and foster research in the area, we describe the following applications of the dataset.

A. Individual vs. Organization

Our dataset contains 1558 repositories developed by 9839 individuals users. As a first application, we extracted repository meta-data from *Github* to get information about the projects. This information is shown in Figure 2. The owners of the projects are classified into two categories: individual user and organization. Among all the repositories, 350 repositories are owned by organizations such as Google, Microsoft, IBM, DeepMind, OpenAi etc. and 1208 repositories are owned by individual developers.

B. API Usage Study

Data science projects in *Python* heavily uses application programming interfaces (API) from common libraries, but the previous research has shown that programmers often struggle to appropriately use APIs. See [16] for a recent work. Some APIs are meant to be called in a specific sequence, with predefined parameter values and guard conditions. Violating these rules will result in a crash, performance degradation or other unwanted output. To identify misuse of an API, we need to identify the good uses of that in the first place. Therefore, we collect top frequent API call sequence patterns from the dataset. In Table III, we have listed the top 10 frequent method call sequences related to neural network. We manually create a list of neural network related methods. For each method M from the list, we search for frequent temporal sequence of k method calls where M is one of the k methods. The result is shown for $k = 8$. The sequence patterns can be leveraged to investigate the violation of those order of API calls.

TABLE III
Top 10 API sequences extracted from the dataset.

API Call Sequences	Count
add, Activation, add, Dropout, add, Dense, add, Activation	115
add, Dense, add, Activation, add, Dropout, add, Dense	114
Dense, add, Activation, add, Dropout, add, Dense, add	112
Conv2d, BatchNorm2d, ReLU, Conv2d, BatchNorm2d, ReLU, Conv2d, BatchNorm2d	103
Sequential, Conv2d, BatchNorm2d, ReLU, Conv2d, BatchNorm2d, ReLU, Conv2d	99
BatchNorm2d, ReLU, Conv2d, BatchNorm2d, ReLU, Conv2d, BatchNorm2d, Lambda	82
Conv2d, BatchNorm2d, ReLU, Conv2d, BatchNorm2d, Lambda, LambdaReduce, ReLU	82
LambdaMap, Sequential, Sequential, Conv2d, BatchNorm2d, ReLU, Conv2d, BatchNorm2d	82
ReLU, Conv2d, BatchNorm2d, ReLU, Conv2d, BatchNorm2d, Lambda, LambdaReduce	82
Sequential, LambdaMap, Sequential, Sequential, Conv2d, BatchNorm2d, ReLU, Conv2d	82

VI. Limitations

Limitations in our dataset can possibly arise from internal and external threats. One internal threat to our dataset could be that projects collected are not representative of data science projects. To alleviate this threat we have used both keyword-based filtering and the use of machine learning libraries as our filtering criteria. A potential external threat could be the lack of maturity of the projects. We use *star* count of at least 80 as a filtering criteria to select repositories to mitigate this threat. Another external threat could be the trustworthiness of the *Python* grammar that we used to parse the programs. We have used the *Python* grammar from official ANTLR [17] website to alleviate this threat.

VII. Summary

Analyzing open source code repositories is a widely used method in programming language and software engineering research. However, no open source dataset for studying data science software is available. We created a dataset from *Github* projects that are using *Python*, a popular programming language for Data Science. The dataset contains 1558 high quality projects with 557,311 revisions. The projects in the dataset are mature, owned by diverse set of user and organizations, and use a large set of machine learning libraries. The dataset has been developed for the *Boa* infrastructure, but it is also available for use outside the *Boa* infrastructure. Finally, we have shown possible research directions to utilize the dataset.

REFERENCES

[1] M. Allamanis and C. Sutton, "Mining source code repositories at massive scale using language modeling," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 207–216.

[2] S. M. Blackburn, R. Garner, C. Hoffmann, A. M. Khang, K. S. McKinley, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S. Z. Guyer *et al.*, "The DaCapo benchmarks: Java benchmarking development and analysis," in *ACM Sigplan Notices*, vol. 41, no. 10. ACM, 2006, pp. 169–190.

[3] I. H. Witten, S. J. Cunningham, and M. D. Apperley, "The new zealand digital library project," *D-Lib magazine*, vol. 2, no. 11, 1996.

[4] H. Do, S. Elbaum, and G. Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact," *Empirical Software Engineering*, vol. 10, no. 4, pp. 405–435, 2005.

[5] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen, "Boa: A language and infrastructure for analyzing ultra-large-scale software repositories," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 422–431.

[6] G. Gousios and D. Spinellis, "GHTorrent: GitHub's data from a firehose," in *Mining software repositories (msr), 2012 9th ieee working conference on*. IEEE, 2012, pp. 12–21.

[7] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble, "The Qualitas corpus: A curated collection of Java code for empirical studies," in *Software Engineering Conference (APSEC), 2010 17th Asia Pacific*. IEEE, 2010, pp. 336–345.

[8] G. Farah, J. S. Tejada, and D. Correal, "Openhub: a scalable architecture for the analysis of software quality attributes," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 420–423.

[9] M. Orrú, E. Tempero, M. Marchesi, R. Tonelli, and G. Destefanis, "A curated benchmark collection of python systems for empirical studies on software engineering," in *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering*. ACM, 2015, p. 2.

[10] R. Dyer, H. Nguyen, H. Rajan, and T. N. Nguyen, "Boa: Ultra-large-scale software repository and source-code mining," *ACM Trans. Softw. Eng. Methodol.*, vol. 25, no. 1, pp. 7:1–7:34, 2015.

[11] *Apache Software Foundation. Hadoop: Open source implementation of MapReduce*, 2014, http://hadoop.apache.org/.

[12] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters 6th symposium on operating system design and implementation," *San Francisco*, 2004.

[13] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, p. 4, 2008.

[14] H. Rajan, T. N. Nguyen, R. Dyer, and H. A. Nguyen, "Boa website," http://boa.cs.iastate.edu/, 2015.

[15] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[16] T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, and M. Kim, "Are code examples on an online Q&A forum reliable?: a study of API misuse on stack overflow," in *Proceedings of the 40th International Conference on Software Engineering*. ACM, 2018, pp. 886–896.

[17] T. J. Parr and R. W. Quong, "ANTLR: A predicated-LL (k) parser generator," *Software: Practice and Experience*, vol. 25, no. 7, pp. 789–810, 1995.