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 1,558 mature Github projects that develop 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 two initial applications 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 prevalent in data-intensive software projects. Among various languages used for Data Science, Python has become one of the most popular languages because of its large collection of libraries to organize and analyze 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], 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. Also, there are some datasets that are not open-source for the community. Another category of datasets includes 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 query on all the revisions of Github projects. While most of the datasets have focused on C, C++ and Java, a few datasets also contain 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, forall, 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 main goal for generating this dataset is to enable MSR research on Data Science programs. For dataset generation, using the GitHub REST APIs, first, we have filtered the projects that use 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 have also filtered projects that have at least 80 stars. By following these filtering criteria, we have ensured that the dataset incorporates high-quality 1,558 Data Science repositories from Github written in Python language. We have
also extended the data schema used by Boa infrastructure, originally designed for Java-like languages, to support Python and written data conversion tools to convert 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 existing ones. Python is also a very well-maintained language that is evolving quickly. Our dataset will help researchers to 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 1,558 Github open-source projects with 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. Besides source code, Github also provides metadata about the project such as developer information, commit date, commit log, etc. that help to answer a larger set of research questions. We have filtered repositories, as discussed below, 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 metadata and source code. 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. For each project, we have collected the metadata 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 have filtered the projects having Data Science related keywords such as machine learning, deep learning, neural network, image processing, artificial intelligence, etc. in the Github description of each repository. However, 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. Finally, we have collected JSON files for 1,558 projects. These JSON files are the input to our dataset generation process.

C. Data Generation

For each chosen Github repository, the data generation starts with caching all the metadata 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 SequenceFiles. Protocol Buffer is an extensible mechanism developed by Google for 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, key is the project and 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 traverse each revision of a Python file and parse it 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 AST format is defined in a flexible way so that it can incorporate most of the AST nodes.
from different languages. However, Python has some different language features 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 modified the Method node in Boa so that it can hold Methods recursively.

E. Data Storage

The storage strategy is the same as other dataset storage mechanism in 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 1,558 repositories developed by 9,839 developers. The projects are owned by both organizations and individual users. Some of the top rated projects are Tensorflow Models, Keras, Scikit-learn, Pandas, Spacy, Spotify Luigi, NVIDIA FastPhotoStyle, Theano, etc. The dataset also contains projects that use at least 33 Data Science libraries including Pytorch, Caffe, Keras, Tensorflow, XGBoost, NLTK, StatsModels, etc. Table I presents the important metrics of the dataset.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>All repositories</td>
<td>1,558</td>
</tr>
<tr>
<td>Developers</td>
<td>9,839</td>
</tr>
<tr>
<td>Revisions</td>
<td>557,311</td>
</tr>
<tr>
<td>Python files (latest snapshot)</td>
<td>86,321</td>
</tr>
<tr>
<td>Python files (all revisions)</td>
<td>4,977,680</td>
</tr>
</tbody>
</table>

B. Data Schema

The dataset is created under the predefined data format of Boa. The fields in the dataset for storing metadata about the repository are shown in Table II.

<table>
<thead>
<tr>
<th>Fields</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>id, name, created_date, code_repositories, ...</td>
</tr>
<tr>
<td>Repository</td>
<td>url, kind, revisions</td>
</tr>
<tr>
<td>Revision</td>
<td>id, log, committer, commit_date, files</td>
</tr>
<tr>
<td>Person</td>
<td>username, real_name, email</td>
</tr>
<tr>
<td>File</td>
<td>name, kind</td>
</tr>
</tbody>
</table>

For storing the parsed AST from the source code, the dataset captures the fields listed in Table III. The top-level node which holds a program file is named ASTRoot in Boa. The Namespace node holds the qualitative path to the file, Declarations and other source code. A class Declaration holds the Python functions as Methods which in turn holds other Statements and Expressions. The fields with kind attribute are the union of different record structures where kind is an enumerated type.

<table>
<thead>
<tr>
<th>Fields</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASTRoot</td>
<td>imports, namespaces</td>
</tr>
<tr>
<td>Namespace</td>
<td>name, modifiers, declarations</td>
</tr>
<tr>
<td>Declaration</td>
<td>name, kind, modifiers, parents, fields, methods, ...</td>
</tr>
<tr>
<td>Type</td>
<td>name, kind</td>
</tr>
<tr>
<td>Method</td>
<td>name, modifiers, return_type, statements, ...</td>
</tr>
<tr>
<td>Variable</td>
<td>name, modifiers, initializer, variable_type</td>
</tr>
<tr>
<td>Statement</td>
<td>kind, condition, expression, statements, ...</td>
</tr>
<tr>
<td>Expression</td>
<td>kind, literal, method, is_postfix, ...</td>
</tr>
<tr>
<td>Modifier</td>
<td>kind, visibility, other, ....</td>
</tr>
</tbody>
</table>

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. A snapshot of this web-based interface is presented in Figure 2. 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 write their own program to submit a Boa query, like SQL queries embedded in other languages, and retrieve results back for analysis.

Fig. 2. Boa web-based interface
B. Boa Query and Output

The Boa program presented in Figure 2 is for counting the most used Data Science libraries. The Boa program contains an input project $p$ and an output aggregator $\text{topimport}$ (line 2). A depth-first search (DFS) 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 for a specific job. If the job is finished without any compilation/execution error, the result can be found under View Job Output and can be downloaded as text file.

C. Using 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 tasks from scratch to analyze the data as shown in [5]. Another method to use the dataset outside of 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. The instructions and resources of this submission can be obtained from [16].

V. APPLICATIONS

To show the potential of the dataset, we describe the following applications of the dataset.

A. Individual vs. Organization

Our dataset contains 1,558 repositories developed by 9,839 individuals developers. As a first application, we extracted repository metadata from Github to get information about the projects. This information is shown in Table I. The owners of the projects are classified into two categories: organization and individual user. Among all the repositories, 350 repositories are managed by organizations such as Google, Microsoft, IBM, DeepMind, OpenAi, etc. and 1,208 repositories are owned by individual developers.

B. API Usage Study

Data Science projects in Python heavily use APIs from common libraries, but previous research [17] has shown that programmers often struggle to use API appropriately. Some APIs are meant to be called in a specific sequence, with predefined parameter types, values and guard conditions. Violating these rules will result in a crash, performance degradation or other unwanted output. To identify the misuse of an API, we need to identify the good uses in the first place. Therefore, we have collected top frequent API call sequence patterns from the dataset. In Table IV, we have listed the top 10 frequent method call sequences related to artificial neural network. We manually created a list of neural network related APIs. For each API method $M$ from the list, we search for frequent temporal sequence of $k$ API calls where $M$ is one of the $k$ API methods. The result in Table IV is shown for $k = 8$. These sequence patterns can be leveraged to investigate the violation of the order of these API calls.

\begin{table}[h]
\centering
\caption{Top 10 API sequences extracted from the dataset.}
\begin{tabular}{|l|l|}
\hline
API Call Sequences & Count \\
\hline
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, Conv2d, BatchNorm2d, ReLU, Conv2d, BatchNorm2d & 82 \\
ReLU, Conv2d, BatchNorm2d, Lambda, LambdaReduce & 82 \\
Sequential, LambdaMap, Sequential, Sequential, Conv2d, BatchNorm2d, ReLU, Conv2d & 82 \\
\hline
\end{tabular}
\end{table}

VI. LIMITATIONS

Limitations in our dataset can arise from internal and external threats. One internal threat to our dataset could be that the collected projects 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 official ANTLR grammar [18] to alleviate this threat.

VII. SUMMARY

Analyzing open source code repositories is a widely used method in software engineering and programming language research. As of this writing, 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 1,558 high-quality projects with 557,311 revisions. The projects in the dataset are mature, owned by a diverse set of users and organizations, and use a large set of machine learning libraries. The dataset has been developed for the Boa infrastructure and also available outside of the infrastructure. Finally, we have shown possible research directions to utilize the dataset.
REFERENCES


