Modeling Topics on Open Source Apache Spark Repositories

Xiaoran Wang, Leonardo Jiménez Rodríguez, and Jilong Kuang

Samsung Research America
Mountain View, CA, USA
{x.wang, leonardo1.r, jilong.kuang}@samsung.com

Abstract—Since Apache Spark’s first release in 2014, it has burst onto the big data scene and became one of the most popular technologies in the domain. Despite the popularity of the Spark project, there have not been research studies towards learning the current state of Spark usage based on a large dataset of open source repositories. Fundamental questions regarding Spark usage remain unknown.

In this paper, our goal is to learn the Spark usage domains by mining a large number of open source projects. We train a topic model to get topics of Spark applications and show that we can obtain the Spark applications’ business domains, techniques, and algorithms. The domains can be used to provide finer granularity of categories than GitHub’s current collection service. Since Spark supports multiple programming languages, we also discuss the topic differences across the three languages.

Index Terms—Mining software repositories, topic modeling, knowledge repository

I. INTRODUCTION

Several technologies have recently been developed to process and analyze the massive and rapidly growing datasets generated by human or machine interactions. Among these technologies, Apache Spark [1] has emerged as the most popular big data framework since its release on May 30th, 2014. Spark, initially developed at the University of California, Berkeley’s AMPLab [2], is an open-source and general purpose cluster-computing framework. It provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Based on Resilient Distributed Dataset (RDD), Spark supports multiple programming languages and provides APIs for various kinds of big data tasks.

Despite its popularity, there has not been any research targeting an overall picture of its current state of usage based on a large number of projects. As such, many questions regarding its usage remain unknown. For example, "what kind of applications is Spark suitable for?", "what programming language fit better for Spark applications in the finance domain?", etc. These questions are often asked when a software engineer needs to decide on technique choices. Actually, the answers to these questions are important not only for Spark application developers but also for Spark API designers. For example, a complete picture of Spark API usage would help its designers understand which business domains heavily use which modules of Spark APIs and get insights of requirements from these business domains. They would therefore meet the developers’ satisfaction and adjust their resources.

To learn the current state of Spark usage, a useful resource is open source projects which use Spark APIs. These projects are Spark applications. As the largest host of source code in the world [3], GitHub is a flagship for current open source development. Such a large repository would be able to provide usage information about Spark. However, due to the number of projects, it is not feasible to manually look into each repository. We need an automatic approach to mine the information from the GitHub repositories.

Latent Dirichlet Allocation (LDA) [4] is an unsupervised machine learning technique which identifies latent topic information. LDA has been widely used on natural language text and became a standard tool in topic modeling. More recently, LDA has also been explored for modeling topics in source code domain [5]–[9]. Because of its popularity and great promise on topic modeling, Binkley et al. [10] conducted a study showing how to tune hyper-parameters for LDA in source code analysis. Despite the prevalence of LDA research, to our knowledge, there has not been studies of topic modeling on Apache Spark applications. Will LDA be able to identify reasonable topics when it is applied to a restricted dataset? What kinds of topics will LDA be able to generate? The answers to these questions are unknown.

In addition to providing invaluable insights for Spark API designers and application developers, topics modeled can be used for tagging or categorizing repositories. As software development often follows the "monkey see monkey do" rule, developers search GitHub for similar applications to quickly learn how to program for their own. GitHub has launched collections 1 which collects hundreds of popular repositories grouped into categories. The collections serves as a catalog that helps developers identify repositories of interest, but it only contains several categories in a coarse level and covers only a small minority of projects. Learning topics in repositories, as we are trying to do in this work, could automate this task and provide more reliable tags or categories.

In this paper, our goal is to obtain insights of the current state of Spark usage by mining open-source corpus from GitHub. Specifically, we are targeting two research questions: (1) What domains is Spark being used in? (2) Are there any differences across programming languages?

1https://github.com/collections
II. APPROACH

In this work, our approach consists of three parts: 1) data collection from GitHub, 2) data preprocessing, and 3) topic modeling using LDA. This section describes the methodologies of each part.

A. Data Collection

In this work, we focus on open source Spark applications from GitHub. We used GitHub API\(^2\) to query the code base with keyword "spark." The results include repositories with the word "spark" in the name, the description, or the README file. We are limiting the results to only repositories where the primary language is Java, Scala, and Python since they are the most common programming languages for using Spark APIs. We also exclude forked projects. Some of these projects are not Apache Spark projects, but projects with keyword "spark" in the name, description or README. So we check each project and see if any Spark API is used in each of the projects. The projects that do not use any Spark APIs are filtered out. In addition, we filter out projects containing less than 5 code files, as these projects are normally small projects for learning or trial purpose.

B. Data Preprocessing

In probabilistic topic modeling, the quality of topics is substantially dependent on the quality of words. It is essential to collect the appropriate words from the source code to build a proper document-word matrix for LDA. Our words collection includes the following four steps:

1) **Tokenization and Keyword Removal.** We tokenize the source code by splitting text using non-word characters. Programming language keywords of each language are filtered, as they do not provide the topic information that we are interested in.

2) **Identifier Splitting.** Identifiers in source code commonly follow the naming convention, such as underscore style and camel casing. Each token from the previous step is split by camel casing character or underscores.

3) **Common Word Removal.** Similar to removing stop words when modeling topics on natural language text, we remove the common words which appear in more than 20% of projects.

4) **Stemming.** After common words removal, we reduce a word to its root form using a traditional stemming algorithm, Porter Stemmer [11].

C. Topic Modeling using LDA

LDA is a technique for deriving probabilistic topic models from textual corpora using a generative process. LDA models each document in a corpus as a mixture of linguistic topics. That is, each document is represented by a probability distribution over the set of linguistic topics inferred by LDA. In doing so, documents are not limited to being associated with a single topic but instead are modeled in a way that considers the possibility that documents may address multiple topics. We consider a project repository as a single document which contains identifiers and comments reflecting business domains and techniques. To train the topic model, we use Mallet [12] which is a state-of-art Java implementation.

**Parameter Tuning.** LDA hyper-parameters include \(\alpha\), \(\beta\) and the number of topics \(K\) [4]. \(\alpha\) is a Dirichlet prior on the per-document topic distribution, while \(\beta\) is a Dirichlet prior on the per-topic word distribution. There is no universal best setting of the LDA parameters, and appropriate settings depend on the problem being solved. We experiment different values of \(\alpha\), \(\beta\) and the number of topics and select a low value of \(\alpha\), so that each repository is composed of only a few topics. Similarly, we choose a low value for \(\beta\) to make each topic have fewer words.

**LDA Metrics.** Let \(d_i\) denote the \(i\)-th document \((1 \leq i \leq N)\) and \(z_k\) denote the topics discovered by LDA \((1 \leq k \leq K)\), where \(N\) and \(K\) stand for the total number of questions and the total number of topics, respectively. Define \(\theta(d_i, z_k)\) as the membership of topic \(z_k\) in document \(d_i\). To obtain the domain topics in research question (1), one way is to count the number of documents whose main topic is \(z_k\). This metric is known as **Number of Documents with Dominant Topic** and was previously defined in [13] as

\[
D(z_k) = \frac{1}{N} \sum_{i=1}^{N} I(d_i, z_k),
\]

where the indicator function is

\[
I(d_i, z_k) = \begin{cases} 
1, & \theta(d_i, z_k) > \theta(d_i, z_l) \ \forall l \neq k \\
0, & \text{o.w.}
\end{cases}
\]

This metric (1) only depends on the dominant topic \(\theta(d_i, z_k)\) disgrading others that are not dominant. To answer our research questions, we are interested in all topics in a fine granularity, we use the **Topic Share** metric from [14], [15]. This metric is given by

\[
\Theta_\delta(z_k) = \frac{1}{N} \sum_{i=1}^{N} \theta_\delta(d_i, z_k),
\]

where

\[
\theta_\delta(d_i, z_k) = \begin{cases} 
\theta(d_i, z_k), & \text{if } \theta(d_i, z_k) \geq \delta \\
0, & \text{o.w.}
\end{cases}
\]

and \(\delta\) is the threshold to remove noise and is set to be 0.05.

III. RESULTS AND DISCUSSION

From time range between May 2014 and November 2017, we collect 12,794 projects. 2,749 (21.5%) of them are Java, 7,889 (61.7%) are Scala, and 2,156 (16.9%) are Python. Figure 1 shows the number of Spark projects created in each month across three different languages since Spark’s official release. Scala not only is the primary language but also has the fastest growth. This figure gives the insights for both Spark API designers and developers when they decide on resource allocation and technique selection.

\(^2\)https://api.github.com/search/repositories
Since Python has grown to be one of the most popular languages in recent years [16], the number which shows Python is much less prevalent than Scala is not what we expected. We check the number of Spark questions tagged with "scala," "java" and "python" from Stack Overflow, and found that Python has more questions than Java. The fact indicates that Python developers may have more issues with Spark APIs.

A. RQ1: Domains of Spark Applications

The first research question is what domains are the Spark applications from (e.g., stock trading, sales, weather, etc.). This question is answered by using LDA topic modeling.

We train a topic model for 2000 iterations with $\alpha = 0.01$, $\beta = 0.01$, and $k = 100$ on the 12,794 spark projects. This topic number allows sufficiently granular topics to be discovered but does not lead to many overlapped topics.

We have two senior software engineers read the 100 topics and annotate the topics. Based on the discussion and agreement, the 100 topics fall into four main categories: business domains, data processing techniques, machine learning/algorithms, and general programming tasks. Table I shows 5 sample topics of each category.

The resulting topics are interesting and indicate that we can use topic models to identify hidden topics of not only business domains but also techniques and even algorithms. When these topics are applied to tag or label the projects, it provides another way of searching the exciting projects.

We examine topics in GitHub collections and compared them with topics from our model. The topics from GitHub collections is in a more coarse level than our topics. For example, the topics include "DevOps tools", "game engines", "music", "software development tools", "software in science", "text editors", "web games", etc. These topics do not contain information about the business domains and specific techniques. Our topics identified by using LDA on Spark projects could be used to enhance the categories in GitHub collection.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Prob</th>
<th>Top Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>0.016</td>
<td>flight, airport, delay, origin, destination</td>
</tr>
<tr>
<td></td>
<td>0.013</td>
<td>sales, customer, price, year, brand</td>
</tr>
<tr>
<td></td>
<td>0.029</td>
<td>day, year, month, hour, temperature</td>
</tr>
<tr>
<td></td>
<td>0.022</td>
<td>latitude, longitude, geo, weather, sensor</td>
</tr>
<tr>
<td></td>
<td>0.032</td>
<td>customer, price, order, stock, product</td>
</tr>
<tr>
<td>Technique</td>
<td>0.017</td>
<td>hbase, family, scan, byte, cell</td>
</tr>
<tr>
<td></td>
<td>0.017</td>
<td>bucket, thrift, protocol, org, channel</td>
</tr>
<tr>
<td></td>
<td>0.038</td>
<td>parquet, hive, meta, commit, props</td>
</tr>
<tr>
<td></td>
<td>0.039</td>
<td>redis, jdbc, connection, mysql, jedis</td>
</tr>
<tr>
<td></td>
<td>0.011</td>
<td>mongo, document, bson, database, uri</td>
</tr>
<tr>
<td>Algorithm</td>
<td>0.034</td>
<td>point, distance, cluster, center, norm</td>
</tr>
<tr>
<td></td>
<td>0.028</td>
<td>token, sentence, pos, sentiment, dictionary</td>
</tr>
<tr>
<td></td>
<td>0.020</td>
<td>graph, edge, vertex, attr, dst</td>
</tr>
<tr>
<td></td>
<td>0.017</td>
<td>tree, statistics, categorical, impurity, decision</td>
</tr>
<tr>
<td></td>
<td>0.010</td>
<td>layer, net, shape, weight, image, loss, network</td>
</tr>
<tr>
<td>Programming</td>
<td>0.008</td>
<td>image, tile, raster, pixel, bounds</td>
</tr>
<tr>
<td>task</td>
<td>0.012</td>
<td>box, color, axis, chart, plot</td>
</tr>
<tr>
<td></td>
<td>0.008</td>
<td>spatial, polygon, geometry, rectangle, grid</td>
</tr>
<tr>
<td></td>
<td>0.027</td>
<td>index, column, histogram, bin, numeric</td>
</tr>
<tr>
<td></td>
<td>0.029</td>
<td>node, tree, child, parent, leaf</td>
</tr>
</tbody>
</table>

B. RQ2: Differences across Programming Languages

To understand the differences across programming languages, Figure 2 shows a heatmap of the average topic assignment for each language, normalized by the maximum average topic assignment. We observe that Scala topic distribution is spread out compared to the other two languages, which indicates that we can evaluate the orthogonality of different languages using cosine similarity like in [17]. Two technologies are orthogonal if they are used to solve different classes of problems. The cosine similarity of the average topic assignment between Scala and Java is 0.53, Scala and Python are 0.46, while Java and Python have a cosine similarity of 0.28 being - unsurprisingly - more orthogonal. From the average topic assignments, we also get insights about the topics that are more common in a language. For example,
"topic modeling" is more common in Scala than in Java and Python, whereas "sentiment analysis" is common in all three languages.

IV. RELATED WORK

The most related work is topic modeling on the source code. Since LDA was introduced by Blei et al. [4], there have been many types of research towards modeling topics from source code [5], [8], [18], [19]. Linstead et al. [5] conducted a study of mining concepts from code with LDA. They treated each source code file as a document and identified seven topics which are related to programming tasks such as logging, database programming, file processing, etc. Similar studies [20], [21] were performed to explore topic models in the source code. Escobar-Avila et al. [22] proposed an unsupervised software categorization using bytecode. In addition, topic modeling along with other methods [23]–[26] has also been applied on different artifacts in software engineering domain, such as forums [27]–[30] and commit messages [29].

V. CONCLUSION

In this work, we use the LDA topic modeling technique to learn the topics embedded in open source Apache Spark repositories. The results show LDA not only generates the business domain topics on this restricted dataset, but also produces topics of techniques, algorithms and programming tasks. These topics are complementary to the manual category produced by GitHub. We also analyze the topic differences among different programming languages by evaluating the orthogonality of different programming languages. The results provide insights for Spark application developers and Spark API designers when they need to decide on technologies and resources.

REFERENCES