Going Big: A Large-Scale Study on What Big Data Developers Ask

Mehdi Bagherzadeh  
*Oakland University*

Raffi T. Khatchadourian  
*CUNY Hunter College*

**How does access to this work benefit you? Let us know!**

Follow this and additional works at: [https://academicworks.cuny.edu/hc_pubs](https://academicworks.cuny.edu/hc_pubs)

Part of the [Artificial Intelligence and Robotics Commons](https://academicworks.cuny.edu/hc_pubs) and the [Software Engineering Commons](https://academicworks.cuny.edu/hc_pubs)

**Recommended Citation**

Going Big: A Large-Scale Study on What Big Data Developers Ask

Mehdi Bagherzadeh
Oakland University
USA
mbagherzadeh@oakland.edu

Raffi Khatchadourian
Hunter College, City University of New York
USA
raffi.khatchadourian@hunter.cuny.edu

ABSTRACT
Software developers are increasingly required to write big data code. However, they find big data software development challenging. To help these developers it is necessary to understand big data topics that they are interested in and the difficulty of finding answers for questions in these topics. In this work, we conduct a large-scale study on Stackoverflow to understand the interest and difficulties of big data developers. To conduct the study, we develop a set of big data tags to extract big data posts from Stackoverflow; use topic modeling to group these posts into big data topics; group similar topics into categories to construct a topic hierarchy; analyze popularity and difficulty of topics and their correlations; and discuss implications of our findings for practice, research and education of big data software development and investigate their coincidence with the findings of previous work.

CCS CONCEPTS
• General and reference → Empirical studies; • Theory of computation → Concurrency.

KEYWORDS
Big data topics, Big data topic hierarchy, Big data topic difficulty, Big data topic popularity, Stackoverflow

ACM Reference Format:

1 INTRODUCTION
“Big data computing is coming to us all” – Fisher et al. [13]

Software developers are increasingly required to write big data code to process and analyze the data that is now abundantly available in a variety of disciplines ranging from science to commerce to national security [11]. For example, a big data software is required to analyze behaviors of large populations of Amazon’s users when interacting with different shopping cart interfaces on its website [13]. Such analysis convinced Amazon to add a feature to its shopping cart that shows personalized recommendations using items in the cart and increase its revenue by 3%, worth hundreds of millions of dollars [25]. Similarly, big data software is required to analyze behaviors of large populations of Zynga’s users when playing online games to allow Zynga for immediate update of their games to create and sell game extras such as a translucent anglerfish [32]. However, developers find big data software development challenging [13–16, 21–24, 26, 37, 42].

To help developers with writing big data software it is necessary to understand the topics that they are interested in and their difficulty in finding answers to questions in these topics. This understanding not only can help these developers and their practice but also the research and education of big data software development by allowing these communities to better decide when and where to focus their efforts. Without such understanding, practitioners may not prepare for similar difficulties, researchers may make incorrect assumptions and educators may teach the wrong topics.

With more than 4 million developer participants and 40 million of their questions and answers, Stack Overflow [39] has become a great source to learn about topics that real world developers are interested in and their difficulties in finding answers to questions in these topics [2, 6, 7, 33, 34, 36, 41].

In this work, we conduct a large-scale study on Stackoverflow to understand the interest and difficulties of big data developers by answering the following research questions:

• RQ1. Big data topics What big data topics do developers ask questions about?
• RQ2. Big data topic hierarchy Can these topics be organized into a hierarchy of topics?
• RQ3. Big data topic popularity What topics are more and less popular among big data developers?
• RQ4. Big data topic difficulty What topics are more and less difficult to find answers to their questions?
• RQ5. Correlation of topic popularity & difficulty How do popularity and difficulty of big data topics correlate?

To answer these questions, we first develop a set of big data tags to extract big data questions and answers from Stackoverflow. Second, we use topic modeling [10] and open card sort [12] to group these questions and answers into big data topics. Third, using the open card sort we group similar topics into categories to construct a topic hierarchy. Fourth, we measure the popularity and difficulty of these topics using several well-known metrics from previous work [2, 6, 7, 31, 36, 36, 40, 41] and analyze their correlations. Finally, we...
discuss the implications of our findings for the practice, research and education of big data software development. We also investigate the coincidence of our findings with the findings of previous work.

A few findings of our study are:

**Big data topics** (1) Big data developers ask questions about a broad spectrum of 28 topics including MapReduce Model, Debugging, Basic Concepts, Performance and Stream Processing. (2) Developers find even the basics of big data software development challenging. (3) Developers ask more about Debugging than Performance.

**Big data topic hierarchy** (4) Big data questions can be grouped into a hierarchy with 9 higher level categories: Programming, Storage, Configuration, Analysis, Debugging, Basic Concepts, Performance, Stream Processing and Logging.

**Big data topic popularity** (5) File Management and Logging are among the most and least popular big data topics.

**Big data topic difficulty** (6) Connection Management and Dataframe are among the most and least difficult big data topics.

**Correlation of topic popularity & difficulty** (7) There is no statistically significant correlation between the popularity and difficulty of big data topics.

**Coincidence of findings with previous work** Some of our findings agree with findings of previous work while others are new.

Our dataset is available at [https://tinyurl.com/y4rw9w2n](https://tinyurl.com/y4rw9w2n).

2 METHODOLOGY

Figure 1 shows an overview of the methodology of our study.

![Diagram](https://example.com/diagram.png)

**Figure 1: An overview of the methodology of our study.**

**Step 1: Download Stackoverflow dataset** In this step of our analysis, we download the Stackoverflow dataset $S$ [38]. The dataset $S$ includes Stackoverflow question and answer posts and their metadata. The metadata of a post includes its identifier, its question or answer type, title, body, tags, creation date, view and favorite counts, score and the identifier of the accepted answer for the post if the post is a question. An answer to a question is an accepted answer if the developer who posted the question marks it as accepted. A question can have 1 to 5 tags.

Our dataset $S$ includes 42,850,541 questions and answers posted over 10 years during August 2008 to December 2018 by 4,142,516 developer participants of Stackoverflow. Among these posts 16,942,340 (39.5%) are questions and 25,908,201 are answers and among these answers 8,917,300 (34.4%) are marked as accepted answers.

**Step 2: Develop big data tag set** In this step of our analysis, we develop a set of big data tags $T$ to identify and extract big data questions in Stackoverflow. First we start with a tag set $T_{init}$ that includes our initial big data tags. Second, we extract questions $P$ from our dataset $S$ whose tags match a tag in $T_{init}$. Third, we construct a set of candidate tags $T$ by extracting tags of questions in $P$. Finally, we refine $T$ by keeping tags that are significantly relevant to big data and exclude others. We use two heuristics $\mu$ and $\nu$ from previous work [41] to measure the significance and relevance of a tag $t$ in the big data tag set $T$.

\[
\begin{align*}
\text{(Significance)} & \quad \mu = \frac{\# \text{of questions with tag } t \text{ in } P}{\# \text{of questions with tag } t \text{ in } S} \\
\text{(Relevance)} & \quad \nu = \frac{\# \text{of questions with tag } t \text{ in } P}{\# \text{of questions with tag } t \text{ in } S}
\end{align*}
\]

A tag $t$ is significantly relevant to big data if its $\mu$ and $\nu$ are higher than specific thresholds. Our 36 experiments with a broad range of threshold values $\mu = (0.05, 0.1, 0.15, 0.2, 0.25, 0.3)$ and $\nu = (0.005, 0.01, 0.015, 0.02, 0.025, 0.03)$ show that $\mu = 0.2$ and $\nu = 0.05$ allow for a significantly relevant set of big data tags. These threshold value are inline with values used in previous work [2, 41].

We use $T_{init} = \{\text{hadoop}\}$ to construct the tag set $T_{\text{Hadoop}}$ for tags related to Hadoop and services in its ecosystem using the above tag development approach. Similarly, we use $T_{init} = \{\text{apache-spark}\}$ to construct $T_{\text{Spark}}$ for tags related to Spark and its ecosystem. Our final big data tag set $T$ is the union of $T_{\text{Hadoop}}$ and $T_{\text{Spark}}$.

\[
T_{\text{Hadoop}} = \{\text{amazon-emr, ambari, apache-pig, bigdata, cloudera, cloudera-cdh, elastic-map-reduce, emr, flume, hadoop, hadoop-partitioning, hadoop-streaming, hadoop2, hbase, hdfs, hdinsight, hive, hiveql, hue, impala, mahout, hortonworks-data-platform, mapreduce, oozie, sqoop, yarn}\}
\]

\[
\]

$T = T_{\text{Hadoop}} \cup T_{\text{Spark}}$

We choose hadoop and apache-spark tags as our initial tags because Hadoop and Spark are industry’s standard ecosystems for big data software development to date used by companies such as Amazon, eBay, Twitter, LinkedIn, IBM, Adobe and NASA JPL. Also Hadoop and Spark are large ecosystems each containing several services for big data software development. Our big data tag set $T$ is very broad and covers a large spectrum of services in Hadoop and Spark ecosystems used in big data software development. A few of these services are Ambari for cluster management, Flume for log analysis, Mahout and Apache-spark-ml for machine learning, Oozie for workflow management, Spark-streaming for stream processing and Yarn for resource management. Our tag set $T$ also includes generic tags such as bigdata.
There are 46 tags in $T$. Table 1 shows tag sets for select significance $p$ and relevance $v$ values.

**Step 3: Extract big data posts** In this step, we use our big data tag set $T$ to extract our set $B$ of big data questions and answers from Stackoverflow. A Stackoverflow question is a big data question if it has a tag that belongs to $T$. The preliminary set of our big data posts includes 112, 948 question and 128,518 answers among these answers 44,577 (34.7%) are accepted answers. To construct $B$, we select questions and their accepted answers from this preliminary set. To reduce noise, following previous work [2, 7, 36], we do not include unaccepted answers in $B$. The final set of big data $B$ includes 157,525 Stackoverflow questions and answers of which 112, 948 (71.7%) are questions and 44,577 (28.3%) are accepted answers.

**Step 4: Preprocess big data post set** In this step, we preprocess the set of big data posts $B$ to reduce the noise [2, 7, 41] for the next step of the analysis. First, we remove code snippets, marked with (code)/ and code, HTML tags, such as paragraph and url tags (p)/ (p) and (a)/ (a), stop words, such as “a”, “the” and “is”, numbers, punctuation marks and non-alphabetical characters. We use MALLET’s list of stop words [28]. Second, we use Porter stemmer [35] to reduce words to their stemmed representations. For example “configuration”, “configure” and “configured” all reduce to “config”.

**Step 5: Model and label big data topics** In this step, we use MALLET [30] with latent Dirichlet allocation (LDA) [10] topic modeling to group our big data posts $B$ into topics. In our topic model, a big data post is a probabilistic distribution of several topics with different proportions. A topic is a set of frequently co-occurring words that approximates a real-world concept. MALLET groups posts into $K$ topics after $I$ grouping iterations and returns a set of topics and their proportions for each post and a set of words for each topic. A post has a dominant topic that covers the biggest proportion of the post.

Our 40 experiments with a broad range of values $K = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$ and $I = \{500, 1000, 1500, 2000\}$ show that $K = 30$ and $I = 1,000$ allow for sufficiently granular topics for our big data posts $B$. These values are inline with values used by previous work [2, 6, 36, 41]. Smaller values of $K$ allow for undesirable combination of multiple unrelated topics into a single topic and large values of $K$ allow for breaking down of a single topic into multiple similar topics. For example, with $K = 10$ MALLET undesirably combines two unrelated Memory Management and Stream Processing topics into a larger topic and with $K = 40$ it breaks down the Dependency Management topic into two smaller topics. MALLET uses hyperparameters $\alpha$ and $\beta$ to control the distribution of words in topics and distributions of topics in posts. We use standard values $50/K$ and 0.01 for these hyperparameters, following previous work [2, 6, 7, 9, 36, 41], because Bigger et al. [9] show that “$\alpha$ and $\beta$ have little influence on the accuracy of the LDA [topic modeling]”.

MALLET groups big data posts in $B$ into topics $t$ that are a set of words but cannot decide about the meaning of a topic and label the topic with a name useful for humans. To illustrate, the word set $w = \{model, spark, vector, mahout, train, feature, data, matrix, algorithm, user, mlib, predict, recommend, cluster, item, set, dataset, label, similar, code\}$ represents a topic in MALLET. Following previous work [2, 6, 7, 31, 36, 41], we use an open card sort [12] to label topics. In open card sort, there is no predefined topics and topics are developed during the labeling process. To label a topic we manually inspect the top 20 words in the set of words for the topic and read through 15 random posts for that topic to come up with a label that best explains words and posts of the topic. To illustrate, we label the word set $w$ with Machine Learning topic. Number of topic words and random posts are inline with previous work [2, 36, 41]. Table 2 shows our 30 big data topics, their names and top 10 stemmed words. We merge topics 10 and 24 into Debugging and topics 27 and 28 into MapReduce Model due to similarities between their words and questions. Our final set of big data topics includes 28 topics. Step 5 answers the research question RQ1.

During the card sort process, the authors individually assign labels to topics and reiterate and refine labels until they agree on topic labels. A third non-author collaborator reviews topic labels, provides suggestions and approves the labels after integration of his suggestions. The first author is a Software Engineer and Programming Languages professor with extensive expertise in concurrent systems of which big data systems are an instance of. The second author is a Software Engineer professor with extensive expertise in streaming and parallel systems. The authors also have several years of industrial experience working as Software Engineers. The non-author collaborator is a Ph.D. Senior Software Engineer at Google with extensive research and programming expertise in concurrent and big data systems, especially FlumeJava.

**Step 6: Construct big data topic hierarchy** In this step, we use a similar card sort process to construct a hierarchy of big data topics by repeated grouping of similar topics into lower and higher level categories. To illustrate, DataFrame and RDD are grouped into the lower level category Dataset. Lower level category Dataset itself is grouped with Primitive category into Datatype which itself is grouped with MapReduce Model, Languages and General Programming into the higher level category Programming. Table 2 and Figure 3 show the hierarchy of big data topics textually and graphically. Step 6 answers the research question RQ2.

**Step 7: Determine big data topic popularity** We use 3 well-known metrics used by previous work to measure the popularity of a big data topic. The first metric is the average number of views for all the questions of a topic [6, 31, 36, 41]. The number of views includes both registered users and visitors of Stackoverflow. The inclusion of views by visitors is important because in Stackoverflow there are many more visitors than registered users [29]. The second metric is the average number of questions of a topic that are marked by users as a favorite question [6, 31, 34, 41]. The third metric is the average score of questions of a topic [6, 31, 34, 41]. Intuitively, a topic with higher number of views and favorites and a higher score is more popular. Table 3 shows the popularity of our big data topics. Step 7 answers the research question RQ3.

**Step 8: Determine topic difficulty** We use 2 well-known metrics used by previous work to measure the difficulty of a big data topic. The first metric is the percentage of questions of a topic that have no accepted answers [36, 40, 41]. The second metric is the average median time needed for questions of a topic to receive accepted answers [36, 41]. Intuitively, a topic with less accepted answers received in longer amount of time is more difficult. Table 4 shows the difficulty of our big data topics. Step 8 answers the research question RQ4.

**Step 9: Determine correlation of topic popularity & difficulty** In this step, we use Kendall correlation to identify if there is
Table 1: $\textit{Hadoop}$ and $\textit{Spark}$ tag sets for select relevance and significance threshold values. Final tag sets are in gray. 

<table>
<thead>
<tr>
<th>(µ, ν)</th>
<th>$\textit{Hadoop}$: Set of tags for Hadoop</th>
<th>No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.2, 0.015)</td>
<td>apache-pig bigdata cloudera hadoop hadoop-streaming hadoop2 hbase hdfs hive hiveml hortonworks-data-platform mapreduce oozie sqoop yarn</td>
<td>15</td>
</tr>
<tr>
<td>(0.2, 0.01)</td>
<td>apache-pig bigdata cloudera-clhd flume hadoop hadoop-streaming hadoop2 hbase hdfs hive hiveml hortonworks-data-platform mahout mapreduce oozie sqoop yarn</td>
<td>18</td>
</tr>
<tr>
<td>(0.2, 0.005)</td>
<td>amazon-emr ambari apache-pig bigdata cloudera cloudera-clhd elastic-map-reduce emr flume hadoop hadoop-partitioning hadoop-streaming hadoop2 hbase hdfs hdfs-hdinsight hive hiveml hortonworks-data-platform hue impala mahout mapreduce oozie sqoop yarn</td>
<td>26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(µ, ν)</th>
<th>$\textit{Spark}$: Set of tags for Spark</th>
<th>No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.2, 0.015)</td>
<td>apache-spark apache-spark-mllib apache-spark-sql pyspark pyspark-sql rdd spark-dataframe spark-streaming yarn</td>
<td>9</td>
</tr>
<tr>
<td>(0.2, 0.01)</td>
<td>apache-spark apache-spark-dataset apache-spark-ml apache-spark-mllib apache-spark-sql parquet pyspark pyspark-sql rdd spark-cassandra-connector spark-dataframe spark-streaming spark-structured-streaming yarn</td>
<td>14</td>
</tr>
<tr>
<td>(0.2, 0.005)</td>
<td>databricks emr parquet pyspark pyspark-sql rdd spark-cassandra-connector spark-dataframe spark-graphx spark-streaming spark-structured-streaming spark-submit sparklyr sparkr yarn</td>
<td>23</td>
</tr>
</tbody>
</table>

3 RESULTS

In this section, we discuss the results of our study for research questions $RQ1$-$RQ5$. We also investigate the coincidence of our findings with the findings of previous work.

3.1 RQ1: Big Data Topics

Table 2 shows the big data topics that developers ask questions about on Stackoverflow, determined in step 5 of our study. According to Table 2, developers ask questions about a broad spectrum of 28 big data topics including MapReduce Model, Debugging, Basic Concepts, Performance and Stream Processing.

The meaning of a big data topic is best understood by looking at questions that belong to the topic. To illustrate, the following is a question with Stackoverflow identifier 2499585 in MapReduce Model topic in which the developer is asking about chaining MapReduce jobs where the output of the reduce phase in a preceding job in the chain is the input of the mapper phase in the following job. This question has been viewed 72,825 times, more than 33 times the average number of views for a Stackoverflow question. The average number of views for a Stackoverflow question is 2,154.

Q. 2499585 **Chaining multiple MapReduce jobs in Hadoop** In many real-life situations where you apply MapReduce, the final algorithms end up being several MapReduce steps. i.e. Map1, Reduce1, Map2, Reduce2, ... So you have the output from the last reduce that is needed as the input for the next map. **What is the recommended way of doing that in Hadoop?**

MapReduce is a programming model for parallel processing of big data in a cluster of distributed machines. In MapReduce, a program is divided into map and reduce functions that map data into a collection of key and value pairs and reduce these pairs by their keys. MapReduce is the programming model for Hadoop and Spark.

Similarly, the following is a question with 12,532 views in Machine Learning topic in which the developer is asking about automatic validation of a classifier built using Random Forest. Random Forest is a learning and classification technique that constructs a set of decision trees during its training and outputs the class that is the mode of the classes output by these trees.

Q. 32769573 **How to cross validate RandomForest model?** I want to evaluate a random forest being trained on some data. Is there any utility in Apache Spark to do the same or do I have to perform cross validation manually?

Finally, the following is a question with 13,444 views in Logging topic in which the developer is asking about batching up and moving 90 seconds worth of logging information into a file in Hadoop Distributed Filesystem (HDFS), using Flume NG. HDFS is the Hadoop file system for distributed storage and retrieval of big data files in a cluster of machines. Flume NG is a service for efficient collection, aggregation and moving of log data.

Q. 14141287 **Using an HDFS Sink and rollInterval in Flume-ng to batch up 90 seconds of log information** I am trying to use Flume-ng to grab 90 seconds of log information and put it into a file in HDFS. I have flume working to look at the log file .. however it is creating a file every 5 seconds instead of .. every 90 seconds.

Finding 1: Developers ask questions about a broad spectrum of 28 big data topics including MapReduce Model, Debugging, Basic Concepts, Performance and Stream Processing.

Figure 2: Big data topics and number of their questions.
The average number of questions for a big data topic is about 4,000. This observation coincides with observations of several previous work and may partly explain the recent focus of previous work on making debugging of big data software rather challenging. This observation shows that developers ask the most questions about MapReduce. This may show that big data developers find MapReduce programming challenging. MapReduce makes big data programming easier by abstracting away details about fault tolerant distribution of both computation and data over a cluster of machines. However, developers still need to understand how to divide and combine computations using mappers and reducers and compose mappers and reducers into complex computational logics over key and value pairs. MapReduce is the dominant programming model for big data programming.

In MapReduce Model topic, developers ask questions like “What is the purpose of shuffling and sorting phase in the reducer in Map Reduce Programming?” and “Simple explanation of MapReduce?”.

Debugging In Figure 2, Debugging is the third topic in the number of questions that developers ask. This may show that developers find debugging of big data software rather challenging. This observation coincides with observations of several previous work and may partly explain the recent focus of previous work on making debugging of big data software easier. Fisher et al. [13] interview 16 data analyst at Microsoft and observe that “a cloud-based computing solution will often be far more difficult to debug.” Kim et al. [23] interview 16 data scientist at Microsoft and observe that some of their participants work on “fault localization and root cause analysis” in debugging. Gulzar et al. [15] propose BigDebug for interactive debugging of big data software written in Spark. Similarly Gulzar et al. [16] propose BigSift for automatic identification of the root cause of a debugging error. More details about these previous works can be found in Section 6.

In Debugging, developers ask questions like “Spark Job running on Yarn Cluster java.io.FileNotFoundException: File does not exits , even though the file exits on the master node” and “Hive:FAILED: SemanticException [Error 10001]: Line 1:14 Table not found ‘Field’”. Yarn is a resource manager and Hive allows for SQL-like queries.

Basic Concepts According to Figure 2, developers also ask more than average number of questions about Basic Concepts. This may show that developers find even the basics of big data development challenging. This observation is not specific to big data software and coincides with observations of several previous work for other software. Ahmed and Bagherzadeh [2] study concurrency questions on Stackoverflow and observe that “[concurrency] developers ask the most about basic concepts [of concurrent software development]”. Similarly, Pinto et al. [34] study 250 most popular concurrency questions on Stackoverflow and observe that “most of them [concurrency questions] are related to basic concepts”.

Table 2: Topic labels, categories, separated by \, and top 10 stemmed words of our big data topics.

<table>
<thead>
<tr>
<th>No.</th>
<th>Topic name</th>
<th>Category</th>
<th>Topic words</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Memory Management</td>
<td>Configuration</td>
<td>memory executor task job spark run core node yarn set</td>
</tr>
<tr>
<td>1</td>
<td>Pig</td>
<td>Analysis</td>
<td>pig script oozl job workflow action run error shell apach</td>
</tr>
<tr>
<td>2</td>
<td>Connection Management</td>
<td>Configuration</td>
<td>connect hadoop server instal cloudera configur user cluster run error</td>
</tr>
<tr>
<td>3</td>
<td>Dataset Load &amp; Store</td>
<td>Storage\Transfer</td>
<td>file data spark read json csv parquet load schema write</td>
</tr>
<tr>
<td>4</td>
<td>Data Organization</td>
<td>Datatype\Dataset</td>
<td>parit join data node case oper kei set shuffl order</td>
</tr>
<tr>
<td>5</td>
<td>Scala Spark</td>
<td>Programming\Languages</td>
<td>scala spark org apach java sql appli anonfun rdd iwc</td>
</tr>
<tr>
<td>6</td>
<td>PySpark</td>
<td>Programming\Languages</td>
<td>python pyspark spark zeppelin error run notebook instal packag azur</td>
</tr>
<tr>
<td>7</td>
<td>Dataframe</td>
<td>Datatype\Dataset</td>
<td>column datafram row spark pyspark data valu function join creat</td>
</tr>
<tr>
<td>8</td>
<td>File Distribution</td>
<td>Storage\File System</td>
<td>node hadoop cluster namenod hdfs datanod start run master slave</td>
</tr>
<tr>
<td>9</td>
<td>General Programming</td>
<td>Programming</td>
<td>class function method object call code type pass work creat</td>
</tr>
<tr>
<td>10</td>
<td>Debugging</td>
<td>Debugging</td>
<td>error code run work problem issu except fail spark log</td>
</tr>
<tr>
<td>11</td>
<td>HBase</td>
<td>Storage</td>
<td>hbase tabl row region zookeep client scan phoenix data server</td>
</tr>
<tr>
<td>12</td>
<td>Job Management</td>
<td>Configuration</td>
<td>spark run cluster job apply submit yarn master emr mode</td>
</tr>
<tr>
<td>13</td>
<td>File Format</td>
<td>Storage\File System</td>
<td>file line read input split text block size data process</td>
</tr>
<tr>
<td>14</td>
<td>File Management</td>
<td>Storage\File System</td>
<td>file hdfs directori path hadoop folder local command creat copi</td>
</tr>
<tr>
<td>15</td>
<td>String</td>
<td>Datatype\Primitive</td>
<td>string type field array null column valu convert charact data</td>
</tr>
<tr>
<td>16</td>
<td>Database Import/Export</td>
<td>Storage\Transfer</td>
<td>sqoop cassandra import tabl data connect jdbi mysql databas connector</td>
</tr>
<tr>
<td>17</td>
<td>Stream Processing</td>
<td>Stream Processing</td>
<td>stream spark kafka data messag batch process topic consum read</td>
</tr>
<tr>
<td>18</td>
<td>Basic Concepts</td>
<td>Basic Concepts</td>
<td>data hadoop store process question system databas solut support work</td>
</tr>
<tr>
<td>19</td>
<td>Date&amp;Time</td>
<td>Datatype\Primitive</td>
<td>date time timestamp dai data month year format event record</td>
</tr>
<tr>
<td>20</td>
<td>Logging</td>
<td>Logging</td>
<td>info flume log java org channel apach sink agent sourc</td>
</tr>
<tr>
<td>21</td>
<td>Performance</td>
<td>Performance</td>
<td>data time perform memori process larg size take run million</td>
</tr>
<tr>
<td>22</td>
<td>Text Search</td>
<td>Analysis</td>
<td>index document mongodb queri collect user elasticearch search mongo result</td>
</tr>
<tr>
<td>23</td>
<td>Dependency Management</td>
<td>Configuration</td>
<td>jar hadoop version spark java error run depend file build</td>
</tr>
<tr>
<td>24</td>
<td>Debugging</td>
<td>Configuration</td>
<td>java org apach hadoop run mapr util main reflect</td>
</tr>
<tr>
<td>25</td>
<td>Machine Learning</td>
<td>Analysis</td>
<td>model spark vector mahout train featur data matrix algorithm user</td>
</tr>
<tr>
<td>26</td>
<td>Hive</td>
<td>Analysis</td>
<td>hive tabl query creat data sql select insert partit column</td>
</tr>
<tr>
<td>27</td>
<td>MapReduce Model</td>
<td>Programming</td>
<td>kei group count valu list result word sort output function</td>
</tr>
<tr>
<td>28</td>
<td>MapReduce Programming</td>
<td>Programming</td>
<td>reduc job map hadoop mapreduc mapper output run input task</td>
</tr>
<tr>
<td>29</td>
<td>RDD</td>
<td>Datatype\Dataset</td>
<td>rdd spark transform code function scala map creat datafram oper</td>
</tr>
</tbody>
</table>
In Basic Concepts, developers ask questions like “What is the actual difference between Data Warehouse & Big Data?”.

Finding 3: Developers ask more than average number of questions about Basic Concepts which may show that they find even the basics of big data software development challenging.

Performance Developers ask more than average number of questions about Performance. This observation is not specific to big data software and coincides with observations of several previous work about other software and may also partly explain the focus of some previous work on improving performance. Fisher et al. [13] observe that “choosing architecture [computation platform] based on cost and performance” is a challenge. Kim et al. [23] observe that some of data scientist at Microsoft work on “performance regression”. Li et al. [27] propose a tool for automatic configuration of Hadoop software to optimize performance. Garbervetsky et al. [14] propose a static analysis for performance optimization of big data queries written in SCOPE language.

However, developers ask less questions about Performance than topics such as MapReduce, Debugging and Basic Concepts. This observation coincides with the observation of the work of Ahmed and Bagherzadeh [2] that observes that “developers ask more about correctness of their concurrent programs than performance”.

Finding 4: Developers ask more questions about MapReduce Model, Debugging and Basic Concepts than Performance.

Stream Processing and String The emergence of Stream Processing topic may be a sign of an increased interest in real-time processing of stream data in addition to original batch data processing in big data software development. This observation may partly explain the recent focus on services for efficient and unified processing of batch and stream data such as Apache Storm [5] and Flink [4]. Similarly, the emergence of String topic may show that data that big data developers are still mostly concerned about processing of textual data and not other forms of data such as images.

Finding 5: Questions that big data developers ask can be grouped into a hierarchy with 9 high level categories: Programming, Storage, Configuration, Analysis, Debugging, Basic Concepts, Performance, Stream Processing and Logging.

3.2 Big Data Topic Hierarchy

Figure 3 shows the hierarchy of big data topics, constructed using step 6 of our study. The figure shows big data topics, in gray, and their categories, in white, and percentages of their questions. The hierarchy expands outwards from higher level categories to lower level categories and topics.

According to Figure 3, questions that big data developers ask can be grouped into a hierarchy with 9 high level categories: Programming, Storage, Configuration, Analysis, Debugging, Basic Concepts, Performance, Stream Processing and Logging. More than one third of questions that developers ask are about Programming, one sixth are about Storage and Configuration and the least are about Logging.

Finding 6: Developers ask the most number of questions about Programming category and the least about Logging.

In the following, we discuss each big data higher level category, its lower categories and topics using sample questions.

3.2.1 Programming Category. More than 1 in 3 big data questions belong to Programming category. Programming is concerned about programming models, languages and datatypes that developers use to write big data software. Programming has 4 lower level categories: Datatype, MapReduce Model, Languages and General Programming. More than half of Programming questions are about Datatype, a quarter are about MapReduce Model and the least are about General Programming.

The Datatype category is concerned about composite datasets such as dataframes and resilient distributed datasets (RDD) as well as primitive datatypes such as String and Date&Time. An RDD is an in-memory distributed dataset that can be operated on in parallel by different machines in a cluster. An RDD is immutable and fault tolerant and allows for data organization through partitioning and join. A dataframe is similar except that its data is organized into named columns, similar to a table in a relational database. RDDs and dataframes are the main datasets in Spark. Dataset category includes questions with titles like (Dataframe): “How to change column types in Spark SQL’s DataFrame?”, (RDD): “(Why) do we need to call cache or persist on a RDD” and (Data Organization): “How to define partitioning of DataFrame?”.

Primitive category includes
questions like (String): "Filter spark DataFrame on string contains" and (Date&Time): "Convert timestamp to date in spark dataframe".

The topic of a question above is included inside parentheses.

MapReduce Model is the programming model that developers ask about in Programming category. This is inline with the general understanding that MapReduce is the dominant model for programming big data software. In Languages category, developers ask about popular programming languages for programming big data software including Python, Scala and Java that Scala translates to. Languages includes questions like (PySpark): "How do I get Python libraries in pyspark?" and (Scala Spark): "how to write case with when condition in spark sql using scala". Finally, the General Programming category includes general programming questions like "How to store custom objects in Dataset?" and "How to escape forward slash in java so that to use it in path".

**Finding 7:** 1/3 of big data questions are about Programming.

**Finding 8:** 1/4 of Programming questions are about MapReduce Model, inline with the general understanding that MapReduce is the dominant model for programming big data software.

### 3.2.2 Storage Category

More than 1 in 6 big data questions belong to Storage category. Storage is concerned about distributed storage of data, random and real-time read and write access and transfer of data. Storage has 3 lower level categories: File System, Transfer and HBase. More than half of questions in Storage are about File System, one third are about Transfer and the least are about HBase.

The File System category is about management, distribution and format of files on Hadoop Distributed File System (HDFS) used by both Hadoop and Spark services. File System includes questions like (File Management): "How to copy file from HDFS to the local file system", (File Distribution): "Hadoop Datanodes cannot find NameNode" and (File Format): "Store images/videos into Hadoop HDFS". File System is mainly about HDFS. In HDFS, a file is partitioned and stored in a distributed cluster of nodes for sequential and batch access. A master datanode stores the metadata of the file and worker datanodes store the actual data partitions. Transfer category is concerned about transferring data between filesystem, datatypes and relational databases. Transfer includes questions like: (Dataset Load&Store): "how to export a data dataframe in pyspark to csv [file]" and (Database Import&Export): "How to import tables from sql server through sqoop to hdfs". HBase category is concerned about random and real-time access and querying the filesystem and includes questions like (HBase): "In HBase is there a way for me to get the middle key of a region?". HBase is a NoSQL database that stores its data in HDFS and adds random and real-time read and write access to HDFS. An HBase table divides its rows into regions.

**Finding 9:** 1/6 of big data questions are about Storage.

### 3.2.3 Configuration

More than 1 in 6 big data questions belong to Configuration category. Configuration is about the configuration of constituent components of a big data software including memory, connections between services in a cluster, dependencies and jobs. Configuration has 4 lower level categories: Job Management, Dependency Management, Connection Management and Memory Management. One third of Configuration questions are about Job Management and Memory Management, a quarter are about Connection Management and the least is about Memory Management.

Configuration includes questions like (Job Management): "How to allocate more executors per worker in Standalone cluster mode?", (Dependency Management): "Hadoop 'Unable to load native-hadoop library for your platform' warning", (Connection Management): "Hadoop: Connecting to ResourceManager failed" and (Memory Management): "How to avoid Spark executor from getting lost and yarn container killing it due to memory limit?".

The observation that 1 out of 6 big data questions are about configuration may partly explain the focus of some recent previous work on making configuration of big data software easier. Li et al. [27] propose a tool for adaptive and automatic configuration of big data programs written in Hadoop to optimize performance.

**Finding 10:** 1/6 of big data questions are about Configuration.

### 3.2.4 Analysis

More than 1 in 7 big data questions belong to Analysis category. Analysis is about analyzing big data using higher level query and scripting languages and functionalities like full-text search and machine learning. This is unlike Programming category in which big data is processed using lower level programming languages such as Python, Scala and Java. Analysis has 4 lower level categories: Hive, Machine Learning, Pig and Search. Near half of Analysis questions are about Hive, less than a quarter about Machine Learning and the least are about Text Search.

Analysis includes questions like (Hive): "Compare two tables of data in HIVE", (Machine Learning): "How can I create a TF-IDF [Term Frequency Inverse Document Frequency] for Text Classification using Spark?", (Pig): "Pig - String extraction using regex" and (Text Search): "Generate Solr Indice Using Customized Mapper in MapReduce". Hive is a database with a SQL-like query language HiveQL to analyze data stored in HDFS. Pig is an analysis platform with a scripting language Pig Latin. A high level Pig Latin analysis translates to a lower level MapReduce program to execute. Solr is a service with indexing and full-text search capabilities. TF-IDF is a numerical statistics used as a weighting factor to denote the importance of a word in a document and in the text search.

Our Machine Learning topic coincides with observations of some previous work. Kandel et al. [20] interview 35 business data analyst and observe that machine learning and especially its feature selection and scale are challenging for data analysts. Kim et al. [23, 24] interview data scientist at Microsoft and categorize data scientists into groups including insight provider, modeling specialist, platform builder, polymath and team leaders. Their modeling specialist is a data scientist with "strong background in machine learning, their main task is to build predictive models".

### 3.2.5 Debugging

The Debugging category is about finding and fixing bugs that manifest during the development and execution of big data software. Debugging includes questions like "What is the meaning of EOF exceptions in hadoop namenode connections from hbase/filesystem?" and "How can i debug Hadoop map reduce?".

### 3.2.6 Basic Concepts

The Basic Concepts category is about fundamental concepts underlying big data software development and
its ecosystems and includes questions like "What is the difference between Big Data and Data Mining?", "Difference between Hadoop and Nosql?" and "Difference between HBase and Hadoop/HDFS?"

3.2.7 Performance. The Performance category is about run time of big data software and its speedup and includes questions like "Spark performance for Scala vs Python" and "Getting the count of records in a data frame quickly".

3.2.8 Stream Processing. The Stream Processing category is about real-time processing of stream data and includes questions like "How to convert Spark Streaming data into Spark DataFrame" and "Spark: processing multiple kafka topic in parallel". Kafka is a streaming platform with publish and subscribe capabilities for streams of records stored in topics.

3.2.9 Logging. The Logging category is about generation and processing of execution logs and includes questions like "Filtering log files in Flume using interceptors" and "Controlling logging functionality in hadoop". Shang et al. [37] analyze execution logs to assist with deployment of big data Hadoop software from a test environment to the real world environment.

3.3 RQ3: Popularity of Big Data Topics

Table 3 shows the popularity of big data topics measured using three metrics average number of views, favorites and score, as described in step 7 of our study. Table 3 is sorted by average number of views for topics. A topic with higher number of views, favorites and score is more popular.

According to Table 3, File Management topic has the highest views, fourth highest favorites and sixth highest score whereas Logging has the third lowest views and second lowest favorites and score. Topic popularity measures can be used by practitioners, researchers and educators to tradeoff one topic against another as discussed in Section 4.

Finding 11: File Management topic in Storage category and Logging are among the most and least popular big data topics.

3.4 RQ4: Difficulty of Big Data Topics

Table 4 shows the difficulty of big data topics measured using two metrics percentage of questions with no accepted answers and average median time to get an accepted answer, as described in step 8 of our study. Table 4 is sorted by percentage of questions with no accepted answers. A topic with higher percentage of questions with no accepted answers and higher median time to receive accepted answers is more difficult.

According to Table 4, Connection Management topic has the fourth highest percentage of questions with no accepted answers and third highest time to accepted answer whereas Dataframe has the lowest percentage of questions with no accepted answers and third lowest time to accepted answer. Topic difficulty measures can be used by practitioners, researchers and educators to tradeoff one topic against another as discussed in Section 4.
3.5 RQ5: Popularity & Difficulty Correlation

According to Tables 3 and 4, a topic such as Logging is not only less popular but also more difficult. Such anecdotal evidence could suggest an intuition that there may be a negative correlation between the popularity and difficulty of big data topics where a more difficult topic is less popular. To investigate we calculate 6 correlations between 3 popularity and 2 difficulty metrics of big data topics as discussed in step 9 of our study.

Table 5 shows the coefficients and p-values for these correlations. According to Table 5, although there is a negative correlation between popularity and difficulty measures of big data topics but the correlation is not statically significant at 95% significance level.

Finding 13: There is no statistically significant correlation between popularity and difficulty of big data topics.

4 IMPLICATIONS

The results of our study can not only help big data developers and their practice but also the research and education of big data software development to better decide where to focus their efforts. Members of these communities could use tradeoffs between big data topics as one of several factors they use in their decision making process. One way to tradeoff a big data topic against another is using popularity and difficulty of these topics.

To illustrate, Figure 4 shows the popularity and difficulty of our top 10 popular big data topics, in Tables 3 and 4. For simplicity, we use the average number of views and the percentage of questions without accepted answers as measures for popularity and difficulty of topics. In Figure 4, a topic is represented by a circle where the size of the circle is proportional to the number of questions in the topic, as shown in Figure 2.

Practice According to Figure 4, a developer with interest in big data software development may decide to start their learning on the more popular and less difficult Dataframe and RDD topics in Datatype category before the less popular and more difficult HBase and Dataset Load&Store topics in Storage category. Similarly, a manager of a big data development team may decide to assign a less difficult task related to Dataframe to a less experienced developer and a more difficult task related to HBase to a more experienced developer. Finally, the manager may allocate more time to the more difficult HBase task when assigning the task to a developer and review their work performance than a less difficult Dataframe task.

Research According to Figure 4, an experienced big data researcher may decide to focus their research on the more difficult and less popular Dependency Management in Configuration category compared to the less difficult and more popular File Management in Storage category in the hope of making more contributions in a less crowded area. This decision is further supported by observations that there are more than average number of questions about Dependency Management and 1 in 6 big data questions are about Configuration. Conversely, a research looking to transition anew to the big data research may decide to start their research on the more popular and less difficult Dataframe in Programming category.

Education According to Figure 4, a big data educator may decide to teach the material for the less difficult and more popular Dataframe and RDD topics before the more difficult and less popular HBase and Dataset Load&Store, if there are no dependencies between these topics. The educator may also decide to prepare more material and spend more time in both the class and lab on the more difficult topic HBase than the less difficult topic Dataframe.

Obviously, there are many factors that go into tradeoffs that practitioners, educators and researchers make to decide where to focus their efforts. However, we believe our findings can contribute to inform and improve these decisions.

5 THREATS TO VALIDITY

Some of our decisions during this study could be a threat to its validity. Using tags in general and big data tags in particular to identify and extract big data questions and answers from Stackoverflow is a threat. This is because big data tags may not identify a complete set of big data questions and answers. To minimize this threat, we use well-known techniques [2, 36, 41] along with extensive experiments to develop a set of big data tags that are significantly relevant to big data software development. Using Stackoverflow as a single dataset to study interests and difficulties of big data developers is another threat. This is because Stackoverflow questions and answers may not be a representative set for questions that developers ask and answers that they provide and receive. However, the large number of Stackoverflow questions and answers and participant developers could help mitigate this threat. Manual labeling of topic word sets is another threat. To minimize this threat, we use well-known techniques [2, 6] to label topics using not only their top words but also a set of random questions. Deciding an optimal value for numbers of topics K, iterations I and hyperparameters α and β is another threat. To minimize this threat, we use well-known techniques [2, 6, 7, 36] and default values in addition to extensive experiments to find reasonable values for these variables. Values we have used are all inline with values used by previous works. It is known that determining an optimal value for K is difficult [7]. Metrics used...
to measure popularity and difficulty of big data topics could be another threat. To minimize this threat, we use well-known metrics to measure popularity [2, 6, 36, 41] and difficulty [2, 36, 40, 41].

6 RELATED


Li et al. [27] propose a tool for automatic configuration of Hadoop programs to optimize performance by learning the relationship between the performance and configuration. Garbervetsky et al. [14] propose a static analysis for performance optimization of big data SCOPE queries that use user-defined functions. The analysis identifies the columns of an input table that are unused or unmodified by the query and optimizes the query by pruning away the unused columns and copy unmodified columns directly from input to output. Shang et al. [37] propose an approach to assist with deployment of Hadoop software from a test environment to the real world environment. They assist by detecting anomalies between executions of the software in the test and real world environments through abstraction and analysis of execution logs. Zhou et al. [42] conduct an empirical study on 210 service quality issues of a big data platform at Microsoft to understand their common symptoms, causes and mitigations. They identify hardware faults, system and customer side effects as major causes of quality issues.

**Data Science** Data science is about transformation of data into insight using both big or small amounts of data. Fisher et al. [13] interview 16 data analyst at Microsoft to characterize big data analytics and its challenges. They identify challenges in 5 categories including acquiring data, selection of computation platform based on cost and performance, shaping data into computation platform formats and iterative writing and debugging of code. Kandel et al. [20] interview 35 business data analyst in industry including healthcare, retail, marketing and finance to characterize industrial data analysis and its challenges. They identify challenges in data discovery, shaping, profiling, modeling and reporting data. Harris et al. [18] interview more than 250 data science practitioner to understand how they view their skills, careers and experiences with prospective employers. They group their practitioners into 5 categories Data Businesspeople, Data Creatives, Data Developers and Data Researchers. Begel and Zimmerman [8] survey 203 and 607 software engineers to identify questions that software engineers would like data scientists to investigate. They identify and rate 145 questions in 12 categories ranging from bug measurements to development best practices to software development life cycle.

Kim et al. [23] interview 16 data scientist at Microsoft to understand the role of data scientists in software engineering including their background, problems they work on and working style. They group data scientists into 5 categories insights provider, modeling specialist, platform builder, polymath and team leaders. Their data scientist work on problems ranging from performance regression to fault localization to customer understanding. Kim et al. [24] extend their previous work [23] and survey 793 data scientist at Microsoft to understand their background, problems they work on and working style, challenges and best practices and tools. Their data scientist work on problems ranging from user engagement to software productivity and quality to domain-specific problems and business intelligence. Their challenges are data challenges for quality, availability and preparation, analysis challenges for scale and machine learning and people challenges. They categorize data scientist into 9 categories polymath, data preparer, shaper and analyzer, platform builder, data evangelist, insight actor and moonlighter.

**Topic Modeling** Ahmed and Bagherzadeh [2] use topic modeling to infer concurrency topics on Stackoverflow and study their popularity and difficulty and correlations. They group their topics into a hierarchy with 8 categories. Yang et al. [41] use topic modeling tuned with a genetic algorithm to infer security topics on Stackoverflow and study their popularity and difficulty. They group their topics into 5 categories. Rosen and Shihab [36] use topic modeling to infer mobile development topics on Stackoverflow and study their popularity and difficulty. They categorize questions that developers ask based on platforms for mobile development and the type of questions into why, what and how questions. Bajaj et al. [6] use topic modeling to infer client-side web development topics on Stackoverflow and study interests of developers in these topics and challenges they face when working with these topics. Barua et al. [7] use topic modeling to infer general topics on Stackoverflow. They study relations of questions and answers of their topics and evolution of interests of developers in these topics.

**ACKNOWLEDGEMENTS**

We would like to thank Yuheng Long and Srilakshmi Sripathi.
REFERENCES


