Summer 8-26-2019

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

Mehdi Bagherzadeh
Oakland University

Raffi T. Khatchadourian
CUNY Hunter College

Recommended Citation

Mehdi Bagherzadeh and Raffi Khatchadourian. Going big: A large-scale study on what big data developers ask. In Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '19. ACM, August 2019.

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:
Mehdi Bagherzadeh and Raffi Khatchadourian. 2019. Going Big: A Large-Scale Study on What Big Data Developers Ask. In *Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19), August 26–30, 2019, Tallinn, Estonia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3338906.3338939

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

ESEC/FSE '19, August 26–30, 2019, Tallinn, Estonia
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-5572-8/19/08…$15.00
https://doi.org/10.1145/3338906.3338939

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.

2 METHODOLOGY

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

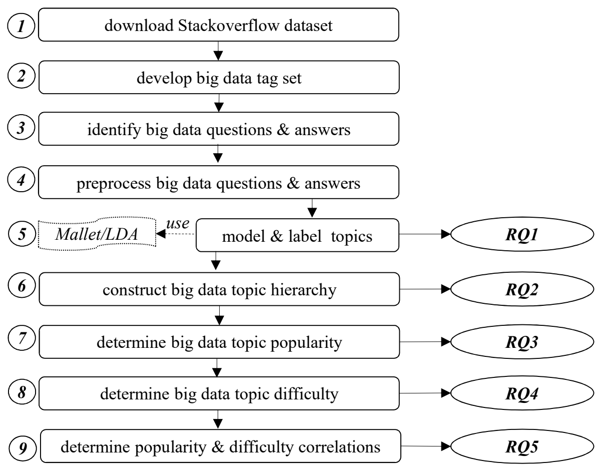
1
download Stackoverflow dataset
2
develop big data tag set
3
identify big data questions & answers
4
preprocess big data questions & answers
5
Mallet/LDA
use
model & label topics
RQ1
6
construct big data topic hierarchy
RQ2
7
determine big data topic popularity
RQ3
8
determine big data topic difficulty
RQ4
9
determine popularity & difficulty correlations
RQ5

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 $\mathcal{S}$ [38]. The dataset $\mathcal{S}$ 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 $\mathcal{S}$ 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 $\mathcal{T}$ to identify and extract big data questions in Stackoverflow. First we start with a tag set $\mathcal{T}_{init}$ that includes our initial big data tags. Second, we extract questions $\mathcal{P}$ from our dataset $\mathcal{S}$ whose tags match a tag in $\mathcal{T}_{init}$. Third, we construct a set of candidate tags $\mathcal{T}$ by extracting tags of questions in $\mathcal{P}$. Finally, we refine $\mathcal{T}$ by keeping tags that are significantly relevant to big data and exclude others. We use two heuristics μ and ν from previous work [41] to measure the significance and relevance of a tag t in the big data tag set $\mathcal{T}$.

$$\text{(Significance)} \quad \mu = \frac{\textit{\# of questions with tag } t \textit{ in } \mathcal{P}}{\textit{\# of questions with tag } t \textit{ in } \mathcal{S}}$$

$$\text{(Relevance)} \quad \nu = \frac{\textit{\# of questions with tag } t \textit{ in } \mathcal{P}}{\textit{\# of questions in } \mathcal{P}}$$

A tag t is significantly relevant to big data if its μ and ν 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 $\nu = 0.005$ 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 $\mathcal{T}_{init} = \{\textit{hadoop}\}$ to construct the tag set $\mathcal{T}_{Hadoop}$ for tags related to Hadoop and services in its ecosystem using the above tag development approach. Similarly, we use $\mathcal{T}_{init} = \{\textit{apache-spark}\}$ to construct $\mathcal{T}_{Spark}$ for tags related to Spark and its ecosystem. Our final big data tag set $\mathcal{T}$ is the union of $\mathcal{T}_{Hadoop}$ and $\mathcal{T}_{Spark}$.

$\mathcal{T}_{Hadoop}$ = {*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*}

$\mathcal{T}_{Spark}$ = {*amazon-emr, apache-spark, apache-spark-2.0, apache-spark-dataset, apache-spark-ml, apache-spark-mllib, apache-spark-sql, apache-zeppelin, emr, databricks, parquet, pyspark, pyspark-sql, rdd, spark-cassandra-connector, spark-dataframe, spark-graphx, spark-streaming, sparklyr, spark-structured-streaming, spark-submit, sparkr, yarn*}

$\mathcal{T} = \mathcal{T}_{Hadoop} \cup \mathcal{T}_{Spark}$

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 $\mathcal{T}$ 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 $\mathcal{T}$ also includes generic tags such as *bigdata*.

There 46 tags in $\mathcal{T}$. Table 1 shows tag sets for select significance μ and relevance ν values.

Step 3: Extract big data posts In this step, we use our big data tag set $\mathcal{T}$ to extract our set $\mathcal{B}$ of big data questions and answers from Stackoverflow. A Stackoverflow question is a big data question if it has a tag that belongs to $\mathcal{T}$. 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 $\mathcal{B}$, 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 $\mathcal{B}$. The final set of big data $\mathcal{B}$ 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 $\mathcal{B}$ to reduce the noise [2, 7, 41] for the next step of the analysis. First, we remove code snippets, marked with $\langle code\rangle\langle/code\rangle$, HTML tags, such as paragraph and url tags $\langle p\rangle\langle/p\rangle$ and $\langle a\rangle\langle/a\rangle$, 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 "configr".

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 $\mathcal{B}$ 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 $\mathcal{B}$. 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 α and β 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 "α and β have little influence on the accuracy of the LDA [topic modeling]".

MALLET groups big data posts in $\mathcal{B}$ 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*, *featur*, *data*, *matrix*, *algorithm*, *user*, *mllib* , *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: $\mathcal{T}_{Hadoop}$ and $\mathcal{T}_{Spark}$ tag sets for select relevance and significance threshold values. Final tag sets are in gray.

(μ, ν)	$\mathcal{T}_{\text{Hadoop}}$: *Set of tags for Hadoop*	*No.*
(0.2, 0.015)	apache-pig bigdata cloudera hadoop hadoop-streaming hadoop2 hbase hdfs hive hiveql hortonworks-data-platform mapreduce oozie sqoop yarn	15
(0.2, 0.01)	apache-pig bigdata cloudera cloudera-cdh flume hadoop hadoop-streaming hadoop2 hbase hdfs hive hiveql hortonworks-data-platform mahout mapreduce oozie sqoop yarn	18
(0.2, 0.005)	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 hortonworks-data-platform hue impala mahout mapreduce oozie sqoop yarn	26
(μ, ν)	$\mathcal{T}_{\text{Spark}}$: *Set of tags for Spark*	*No.*
(0.2, 0.015)	apache-spark apache-spark-mllib apache-spark-sql pyspark pyspark-sql rdd spark-dataframe spark-streaming yarn	9
(0.2, 0.01)	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	14
(0.2, 0.005)	amazon-emr apache-spark apache-spark-2.0 apache-spark-dataset apache-spark-ml apache-spark-mllib apache-spark-sql apache-zeppelin databricks emr parquet pyspark pyspark-sql rdd spark-cassandra-connector spark-dataframe spark-graphx spark-streaming spark-structured-streaming spark-submit sparklyr sparkr yarn	23

a statically significant correlation between popularity and difficulty of our big data topics. We use Kendall because it is less sensitive to outliers and therefore relatively more stable. In total, we investigate 6 correlations between 3 popularity metrics and 2 difficulty metrics for big data topics. Step 9 answers the research question ***RQ5***.

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 Forst 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*.

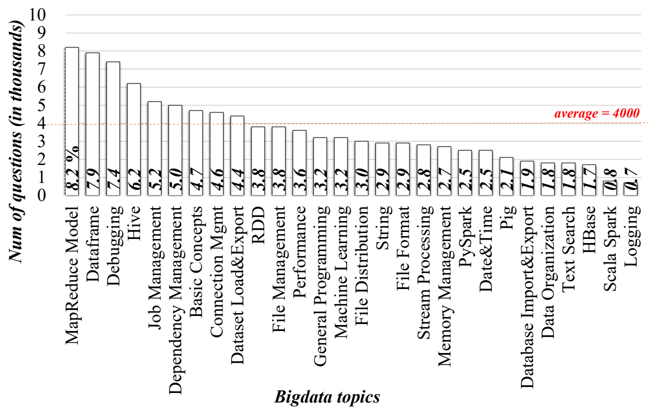

Figure 2: Big data topics and number of their questions.

Figure 2 shows the numbers of questions that developers ask about our big data topics. According to Figure 2, the number of

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

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

questions that developers ask for big data topics are not uniform. Developers ask the most number of questions (8.2%) about *MapReduce Model* and the least number of questions (0.7%) about *Logging*. The average number of questions for a big data topic is about 4,000.

Finding 2: Developers ask the most questions about *MapReduce Model* and the least about *Logging* topic.

MapReduce Model According to Figure 2, developers ask the most questions about *MapReduce Model*. 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 software 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".

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 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.

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

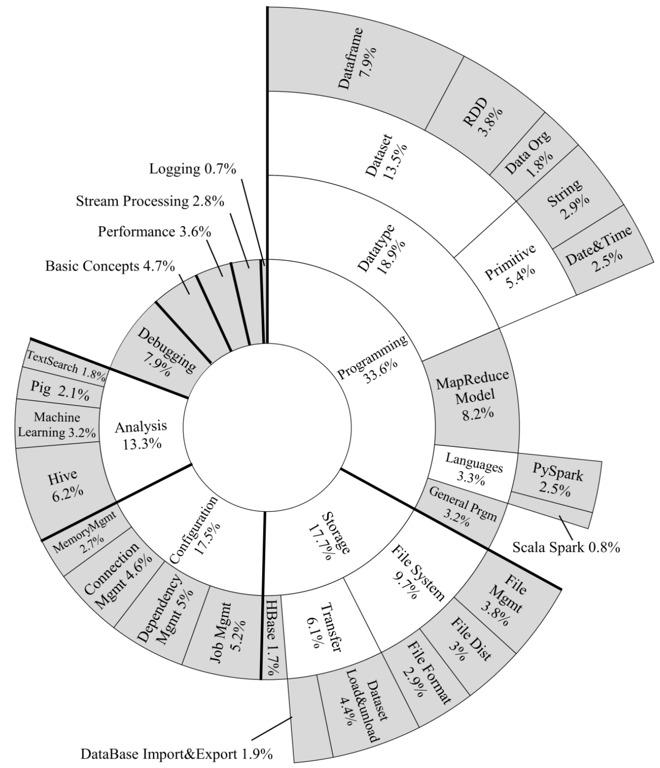

Figure 3: Hierarchy of big data topics, in gray, their categories, in white and percentages of their questions.

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 table 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

Table 3: Popularity of big data topics.

Topic	*Avg. views*	*Avg. favorites*	*Avg. score*
File Management	2141.6	0.5	1.4
File Distribution	1861.6	0.5	1.3
Hive	1811.6	0.3	1.0
Dependency Management	1774.7	0.4	1.4
Dataframe	1731.4	0.5	1.3
String	1600.8	0.3	1.0
RDD	1543.8	0.5	1.6
HBase	1464.9	0.4	1.3
Dataset Load&Store	1410.7	0.4	1.2
Data Organization	1369.3	0.8	2.0
File Format	1315.1	0.5	1.2
Connection Management	1301.7	0.3	1.0
MapReduce Model	1294.2	0.5	1.2
Debugging	1278.3	0.3	1.2
Basic Concepts	1248.7	0.9	1.8
PySpark	1152.5	0.4	1.3
Memory Management	1141.1	0.6	1.7
Job Management	1131.2	0.4	1.5
General Programming	1130.9	0.5	1.6
Pig	1079.8	0.2	0.8
Date&Time	1068.1	0.3	0.7
Text Search	1053.8	0.5	1.3
Scala Spark	1050.0	0.3	1.0
Database Import&Export	1048.7	0.3	0.8
Performance	840.3	0.5	1.4
Logging	820.7	0.3	0.8
Machine Learning	763.9	0.5	1.3
Stream Processing	715.0	0.4	1.2
Big data Average	*1364.0*	*0.4*	*1.3*

Table 4: Difficulty of big data topics.

Topic	*% w/o acc. answer*	*Hrs to acc. answer*
Stream Processing	71.3	7.4
Database Import&Export	69.8	7.9
Memory Management	69.0	7.7
Connection Management	68.7	17.0
PySpark	68.1	7.6
Logging	66.9	22.5
Hive	66.0	3.6
Job Management	65.7	9.2
Scala Spark	64.8	2.6
Debugging	64.3	2.4
Dependency Management	64.2	4.7
Performance	64.0	3.8
Dataset Load&Store	63.5	2.9
HBase	63.4	17.2
File Distribution	63.1	6.7
Pig	63.0	10.1
File Management	62.3	3.7
File Format	61.9	4.0
Machine Learning	61.6	4.6
Basic Concepts	59.8	5.5
Data Organization	58.5	3.2
Text Search	57.5	5.2
Date&Time	55.8	2.1
General Programming	53.8	1.8
String	51.9	1.3
MapReduce Model	51.6	0.2
RDD	49.4	1.1
Dataframe	46.3	1.2
Big data Average	*60.5*	*3.3*

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.

Table 5: Correlations of big data topics popularity/difficulty.

coefficient / p-value	*Avg. views*	*Avg. favorites*	*Avg. score*
% w/o acc. answer	-2.33 / 0.082	-3.26 / 0.026	-1.79 / 0.201
Hrs to acc. answer	-2. 01 / 0.133	-1.57 / 0.283	-0.61 / 0.660

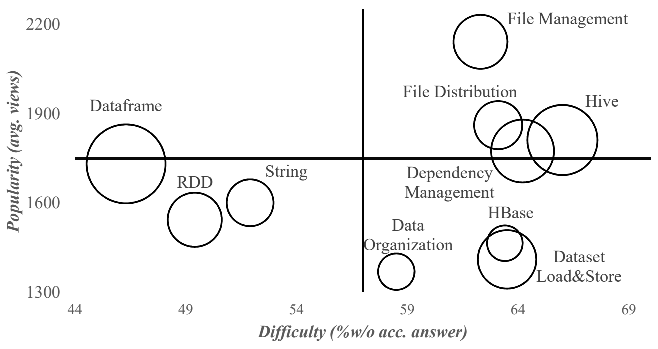

Figure 4: Tradeoff big data topics by popularity/difficulty.

Finding 12: *Connection Management* topic in *Configuration* category and *Dataframe* in *Programming* are among the most and least difficult big data topics.

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 statically 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

Big data Gulzar et al. [15] propose BigDebug for interactive debugging of big data Spark software. BigDebug supports interactive debugging through simulated breakpoints and on demand watchpoints to allow for inspection of a program without pausing its entire computation and retrieval of select intermediate data. Gulzar et al. [16] propose BigSift for automatic identification of the root cause of an error. BigSift uses delta debugging and data provenance to identify the root cause of an error in a Spark program.

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 insight 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.

Gyöngyi et al. [17] and Adamic et al. [1] study Yahoo!Answers questions and answers to determine interests of developers in a set of predefined and fixed categories. Hindle et al. [19] use topic modeling to infer topics related to development tasks from commit messages of a software project and study the evolution of interests in these topics. Treude et al. [40] and Allamanis and Sutton [3] study Stackoverflow questions and answers to infer types of questions that developers ask and determine their difficulties.

In contrast, in this work we extract, topic model and categorize 157,525 big data questions and answers on Stackoverflow to understand big data topics that developers are interested in, the hierarchy of these topics, their popularity, difficulty and their correlations and implications of such understanding for practice, research and education of big data software development.

7 CONCLUSION AND FUTURE WORK

In this work, we conduct a large-scale study on Stackoverflow to understand interest and difficulties of big data developers. We infer big data topics that developers ask questions about, organize them into a hierarchy of topics and measure their popularity and difficulty and correlations. We show the coincidence of findings of our study with findings of previous work. We also show how our findings can help practice, research and education of big data software development. One avenue of future work is to conduct similar large-scale studies using commit logs and bug reports to triangulate with our results.

ACKNOWLEDGEMENTS

We would like to thank Yuheng Long and Srilakshmi Sripathi.

REFERENCES

[1] Lada A. Adamic, Jun Zhang, Eytan Bakshy, and Mark S. Ackerman. 2008. Knowledge Sharing and Yahoo Answers: Everyone Knows Something. In *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*. ACM, New York, NY, USA, 665–674. https://doi.org/10.1145/1367497.1367587

[2] Syed Ahmed and Mehdi Bagherzadeh. 2018. What Do Concurrency Developers Ask About?: A Large-scale Study Using Stack Overflow. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '18)*. ACM, New York, NY, USA, Article 30, 10 pages. https://doi.org/10.1145/3239235.3239524

[3] Miltiadis Allamanis and Charles Sutton. 2013. Why, when, and what: analyzing Stack Overflow questions by topic, type, and code. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 53–56.

[4] Apache Flink. [n.d.]. https://flink.apache.org/.

[5] Apache Storm. [n.d.]. http://storm.apache.org/.

[6] Kartik Bajaj, Karthik Pattabiraman, and Ali Mesbah. 2014. Mining Questions Asked by Web Developers. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014)*. ACM, New York, NY, USA, 112–121. https://doi.org/10.1145/2597073.2597083

[7] Anton Barua, Stephen W Thomas, and Ahmed E Hassan. 2014. What are developers talking about? an analysis of topics and trends in Stack Overflow. *Empirical Software Engineering* 19, 3 (2014), 619–654.

[8] Andrew Begel and Thomas Zimmermann. 2014. Analyze This! 145 Questions for Data Scientists in Software Engineering. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, New York, NY, USA, 12–23. https://doi.org/10.1145/2568225.2568233

[9] Lauren R. Biggers, Cecylia Bocovich, Riley Capshaw, Brian P. Eddy, Letha H. Etzkorn, and Nicholas A. Kraft. 2014. Configuring Latent Dirichlet Allocation Based Feature Location. *Empirical Softw. Engg.* 19, 3 (June 2014), 465–500. https://doi.org/10.1007/s10664-012-9224-x

[10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (March 2003), 993–1022. http://dl.acm.org/citation.cfm?id=944919.944937

[11] C.L. Philip Chen and Chun-Yang Zhang. 2014. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences* 275 (2014), 314 – 347. https://doi.org/10.1016/j.ins.2014.01.015

[12] Sally Fincher and Josh Tenenberg. 2005. Making sense of card sorting data. *Expert Systems* 22, 3 (2005), 89–93.

[13] Danyel Fisher, Rob DeLine, Mary Czerwinski, and Steven Drucker. 2012. Interactions with Big Data Analytics. *interactions* 19, 3 (May 2012), 50–59. https://doi.org/10.1145/2168931.2168943

[14] Diego Garbervetsky, Zvonimir Pavlinovic, Michael Barnett, Madanlal Musuvathi, Todd Mytkowicz, and Edgardo Zoppi. 2017. Static Analysis for Optimizing Big Data Queries. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017)*. ACM, New York, NY, USA, 932–937. https://doi.org/10.1145/3106237.3117774

[15] Muhammad Ali Gulzar, Matteo Interlandi, Tyson Condie, and Miryung Kim. 2016. BigDebug: Interactive Debugger for Big Data Analytics in Apache Spark. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. ACM, New York, NY, USA, 1033–1037. https://doi.org/10.1145/2950290.2983930

[16] Muhammad Ali Gulzar, Siman Wang, and Miryung Kim. 2018. BigSift: Automated Debugging of Big Data Analytics in Data-intensive Scalable Computing. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*. ACM, New York, NY, USA, 863–866. https://doi.org/10.1145/3236024.3264586

[17] Zoltán Gyöngyi, Georgia Koutrika, Jan Pedersen, and Hector Garcia-Molina. 2008. Questioning Yahoo! Answers. In *Proceedings of the 1st workshop on question answering on the Web.*

[18] Harlan Harris, Sean Murphy, and Marck Vaisman. 2013. *Analyzing the Analyzers: An Introspective Survey of Data Scientists and Their Work.* O'Reilly Media, Inc.

[19] A. Hindle, M. W. Godfrey, and R. C. Holt. 2009. What's hot and what's not: Windowed developer topic analysis. In *2009 IEEE International Conference on Software Maintenance*. 339–348. https://doi.org/10.1109/ICSM.2009.5306310

[20] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2012. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (Dec. 2012), 2917–2926. https://doi.org/10.1109/TVCG.2012.219

[21] Raffi Khatchadourian, Yiming Tang, Mehdi Bagherzadeh, and Syed Ahmed. 2018. A Tool for Optimizing Java 8 Stream Software via Automated Refactoring. In *International Working Conference on Source Code Analysis and Manipulation (SCAM '18)*. IEEE, 34–39. https://doi.org/10.1109/SCAM.2018.00011

[22] Raffi Khatchadourian, Yiming Tang, Mehdi Bagherzadeh, and Syed Ahmed. 2019. Safe Automated Refactoring for Intelligent Parallelization of Java 8 Streams. In *International Conference on Software Engineering (ICSE '19)*. ACM/IEEE, NJ, USA, 619–630. https://doi.org/10.1109/ICSE.2019.00072

[23] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2016. The Emerging Role of Data Scientists on Software Development Teams. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 96–107. https://doi.org/10.1145/2884781.2884783

[24] M. Kim, T. Zimmermann, R. DeLine, and A. Begel. 2018. Data Scientists in Software Teams: State of the Art and Challenges. *IEEE Transactions on Software Engineering* 44, 11 (Nov 2018), 1024–1038. https://doi.org/10.1109/TSE.2017.2754374

[25] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M. Henne. 2009. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery* 18, 1 (01 Feb 2009), 140–181. https://doi.org/10.1007/s10618-008-0114-1

[26] Tanakorn Leesatapornwongsa, Jeffrey F. Lukman, Shan Lu, and Haryadi S. Gunawi. 2016. TaxDC: A Taxonomy of Non-Deterministic Concurrency Bugs in Datacenter Distributed Systems. *SIGARCH Comput. Archit. News* 44, 2 (March 2016), 517–530. https://doi.org/10.1145/2980024.2872374

[27] C. Li, H. Zhuang, K. Lu, M. Sun, J. Zhou, D. Dai, and X. Zhou. 2014. An Adaptive Auto-configuration Tool for Hadoop. In *2014 19th International Conference on Engineering of Complex Computer Systems*. 69–72. https://doi.org/10.1109/ICECCS.2014.17

[28] Mallet stop words. [n.d.]. https://github.com/mengjunxie/ae-lda/blob/master/misc/mallet-stopwords-en.txt.

[29] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. 2011. Design Lessons from the Fastest QA Site in the West. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2857–2866. https://doi.org/10.1145/1978942.1979366

[30] Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. (2002). http://mallet.cs.umass.edu.

[31] Sarah Nadi, Stefan Krüger, Mira Mezini, and Eric Bodden. 2016. "Jumping Through Hoops": Why do Java Developers Struggle with Cryptography APIs?. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16).*

[32] Nick Wingfield. September 2011. Wall Street Journal https://www.wsj.com/articles/SB10001424053111904823804576502442835413446.

[33] Gustavo Pinto, Fernando Castor, and Yu David Liu. 2014. Mining Questions About Software Energy Consumption. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014)*. ACM, New York, NY, USA, 22–31. https://doi.org/10.1145/2597073.2597110

[34] Gustavo Pinto, Weslley Torres, and Fernando Castor. 2015. A Study on the Most Popular Questions About Concurrent Programming. In *Proceedings of the 6th Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU 2015)*. ACM, New York, NY, USA, 39–46. https://doi.org/10.1145/2846680.2846687

[35] M. F. Porter. 1997. Readings in Information Retrieval. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Chapter An Algorithm for Suffix Stripping, 313–316. http://dl.acm.org/citation.cfm?id=275537.275705

[36] Christoffer Rosen and Emad Shihab. 2016. What are mobile developers asking about? a large scale study using Stack Overflow. *Empirical Software Engineering* 21, 3 (2016), 1192–1223.

[37] Weiyi Shang, Zhen Ming Jiang, Hadi Hemmati, Bram Adams, Ahmed E. Hassan, and Patrick Martin. 2013. Assisting Developers of Big Data Analytics Applications when Deploying on Hadoop Clouds. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*. IEEE Press, Piscataway, NJ, USA, 402–411. http://dl.acm.org/citation.cfm?id=2486788.2486842

[38] Stack Exchange Dump. June 2017. https://archive.org/details/stackexchange.

[39] Stack Overflow. December 2018. http://www.stackoverflow.com/.

[40] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. 2011. How do programmers ask and answer questions on the web?: Nier track. In *Software Engineering (ICSE), 2011 33rd International Conference on*. IEEE, 804–807.

[41] Xin-Li Yang, David Lo, Xin Xia, Zhi-Yuan Wan, and Jian-Ling Sun. 2016. What Security Questions Do Developers Ask? A Large-Scale Study of Stack Overflow Posts. *Journal of Computer Science and Technology* 31, 5 (01 Sep 2016), 910–924. https://doi.org/10.1007/s11390-016-1672-0

[42] Hucheng Zhou, Jian-Guang Lou, Hongyu Zhang, Haibo Lin, Haoxiang Lin, and Tingting Qin. 2015. An Empirical Study on Quality Issues of Production Big Data Platform. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2 (ICSE '15)*. IEEE Press, Piscataway, NJ, USA, 17–26. http://dl.acm.org/citation.cfm?id=2819009.2819014