UNSTACKING THE OVERFLOW

A CLOSER LOOK AT THE DATA SCIENCE COMMUNITY

PREPARED BY:
RUSH BHARDWAJ, JOSH CHEN,
SAMEEN HAROON, SHARON XU

TABLE OF CONTENTS

Introduction

The goal of this project is to analyze the posts on Stack Overflow to understand popular topics and trends in the programmer community. We used Stack Overflow public data on BigQuery in Google Cloud Platform. We are particularly interested in this topic for several reasons. First, as a team of aspiring data scientists, we use Stack Overflow regularly and have emotional attachment to this friendly community. Secondly, we would like to learn about the tools and skill sets most valued by the industry. The popularity of topics on Stack Overflow might be able to inform us about this trend, letting us better understand our position in the labor market as prospective business data scientists. Lastly, the Stack Overflow dataset is substantial in size, and text processing is computationally expensive so analyzing this dataset using a distributed computing framework is appropriate.

Diving deeper, the objective of this project is to uncover trends and insights related to data science through descriptive analysis of post patterns and text mining to find popular topics in the community. The process includes utilizing the tools in the Google Ecosystem for data accessing and analytics; thus, it has both the tutorial aspect of setting up the tools and the analytics aspect of applying text mining on the title and body of questions. In general, the descriptive analysis mainly used BigQuery (SQL language) and Tableau while text mining was performed on Dataproc clusters using Pyspark.

Relevant technology

Google cloud ecosystem

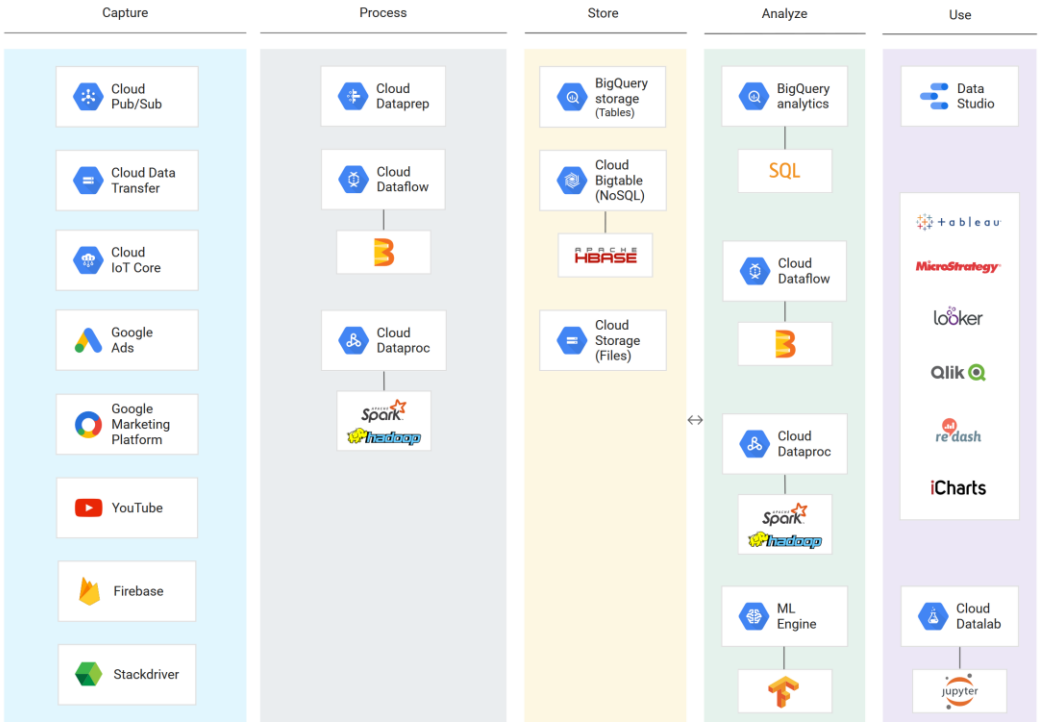


Figure 1: Overview of Google Cloud ecosystem for big data[1]

The Google cloud platform (GCP) is a suite of cloud computing services provided by google using their Google cloud infrastructure, illustrated in Figure 1. GCP is used by Google for in-house applications and systems and external end user products (examples: Spotify, HSBC, Home Depot). GCP's big data solutions are modeled toward optimizing performance, scalability and security. To focus on security, Google uses the same security models as Gmail or Google apps. Google cloud

[1] Image sourced from https://cloud.google.com/solutions/big-data/

computing platform supports many languages like Java, Python, C#, Go, Node.js, PHP, or Ruby. GCP's architecture is designed to reduce cost and for easy integration with open-source tools like Apache's Hadoop, Impala, Spark and Hive. As GCP focuses in providing a complete stop shop outlook to data and its warehousing, it also has pre-developed BI frameworks for marketing analytics, stream analytics and workflow orchestration. For all these reasons, and our limited prior familiarity with the Google ecosystem, we opted to focus our project on Google cloud technology.

Big data tools used

For our exploratory analysis, we relied directly on retrieiving and processing data within the BigQuery user interact, with supporting visualizations in Tableau. However, for machine learning applications, we leveraged relationships between different technologies in the Google ecosystem for data transfer and analysis.

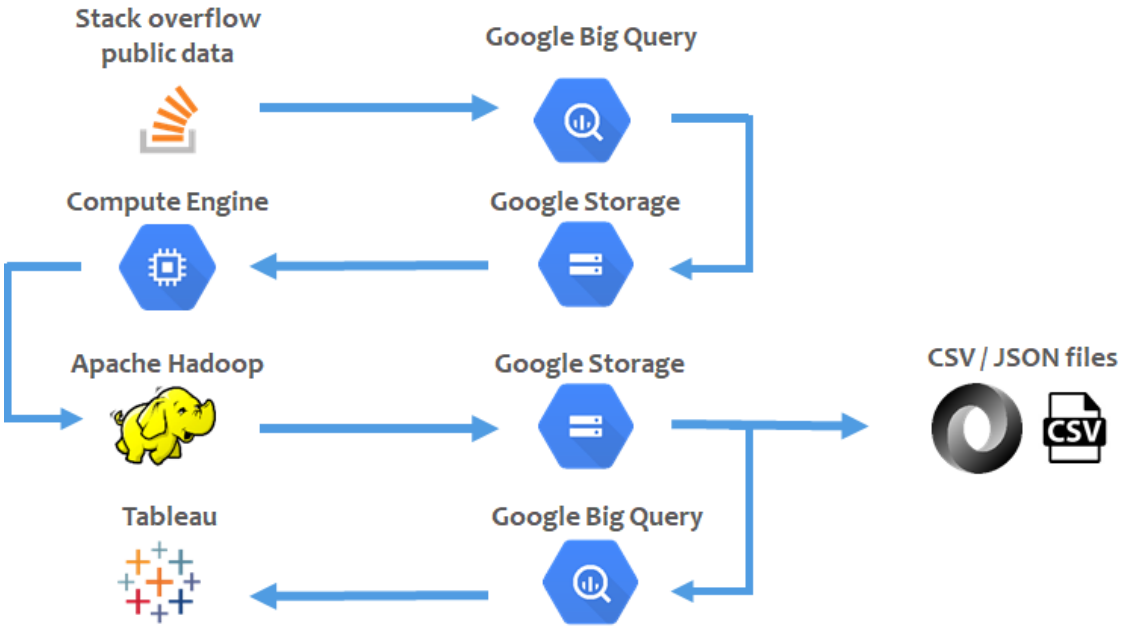


Figure 2: Overview of data transfer process across technologies used

As shown in figure 2, we started our machine learning analysis by acquiring stack overflow posts data from Google Big Query. To analyze the data, a SQL prompt was issued on the GCP big-query console and data was stored as a data table on big-query. Using a PySpark big-query connector, Hadoop was started on a compute engine on GCP and the data table stored in big-query was transferred to a bucket on Goggle cloud's storage platform as a resilient distributed dataset (RDD). For easy computation, the RDD was converted to a data frame. We then processed data and conducted text mining using spark.sql and pyspark.ml, detailed in the section on text mining. The required output was recorded as a data frame and converted to a csv and json file and transferred back to the storage bucket.

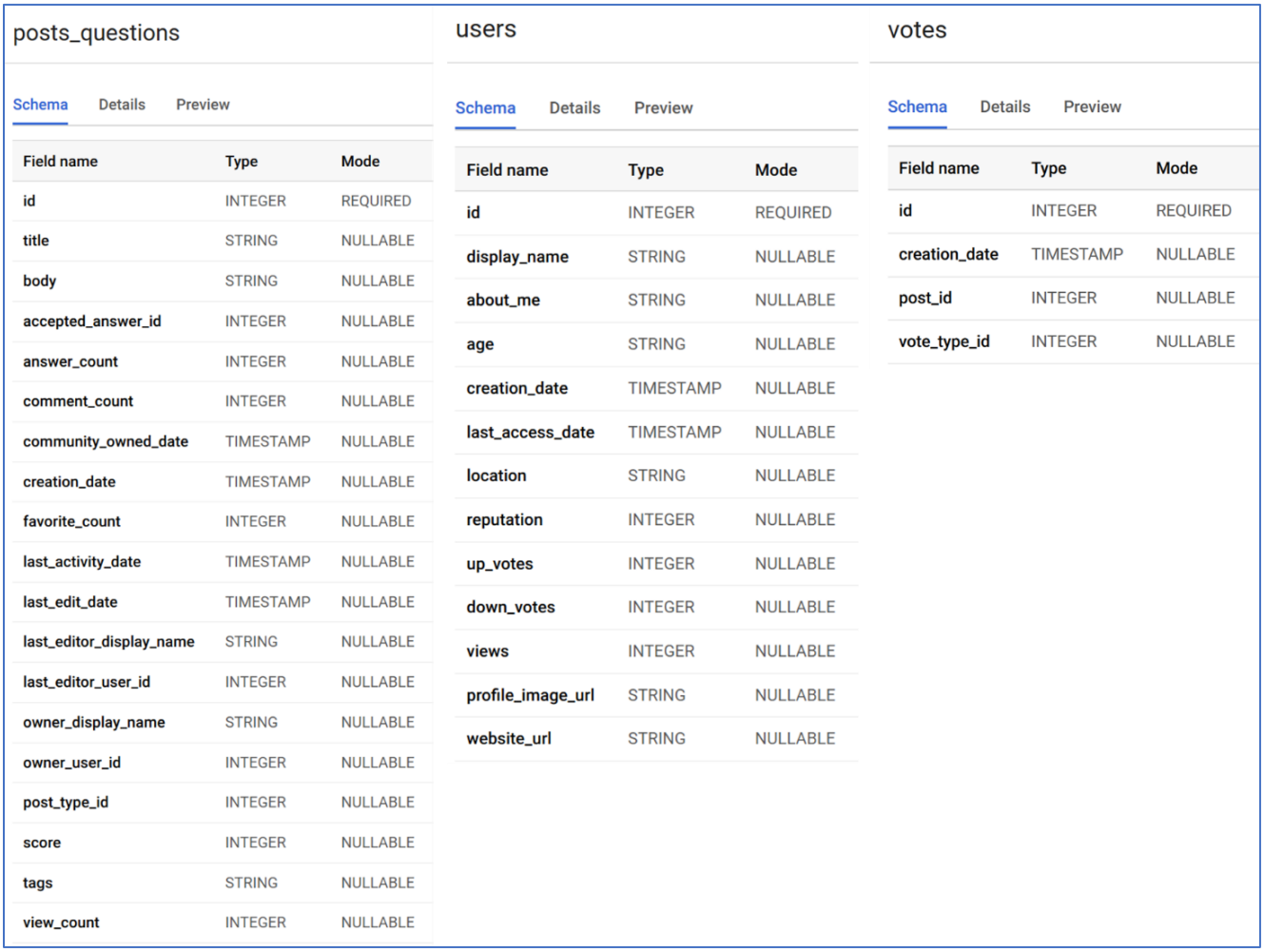
The next step was to take the Universal Resource Identifier (URI) and upload the data back to the big-query platform. To name the stored bucket, two URI for the storage were formulated. One to test a single file upload by specifying the object name containing the data and another with a wildcard in the URI. The wildcard URI helps putting together data separated into multiple files (which share a common base name). While uploading the data through the URIs, a major error was encountered as there were many "," in the location attribute, it became difficult to upload the data on GCP big-query and became a bottle neck in the process. Since our primary goal was to validate the possibility of data transfer between these systems, we ended our exploration here.

Exploratory analysis

Process

The dataset used for this project included Stack Overflow posts written between 2008 and 2018, stored as a database spanning 16 tables with content on posts, comments, users, votes, etc, with a total size of over 200 GB. Standard SQL queries were used to access, process and store relevant subsets of data that enabled descriptive analysis of trends and themes of stack overflow posts.

In order to focus our efforts, we primarily examined questions posted on the forum, looking at measures of volume, engagement (view counts, favorite counts, comment counts), and popularity (measured by percent of upvotes). Three main tables, shown in Figure 3 were used for some basic summary statistics, and to derive the final tables that were used for descriptive insights on data science trends.



posts_questions

Schema | Details | Preview

Field name	Type	Mode
id	INTEGER	REQUIRED
title	STRING	NULLABLE
body	STRING	NULLABLE
accepted_answer_id	INTEGER	NULLABLE
answer_count	INTEGER	NULLABLE
comment_count	INTEGER	NULLABLE
community_owned_date	TIMESTAMP	NULLABLE
creation_date	TIMESTAMP	NULLABLE
favorite_count	INTEGER	NULLABLE
last_activity_date	TIMESTAMP	NULLABLE
last_edit_date	TIMESTAMP	NULLABLE
last_editor_display_name	STRING	NULLABLE
last_editor_user_id	INTEGER	NULLABLE
owner_display_name	STRING	NULLABLE
owner_user_id	INTEGER	NULLABLE
post_type_id	INTEGER	NULLABLE
score	INTEGER	NULLABLE
tags	STRING	NULLABLE
view_count	INTEGER	NULLABLE

users

Schema | Details | Preview

Field name	Type	Mode
id	INTEGER	REQUIRED
display_name	STRING	NULLABLE
about_me	STRING	NULLABLE
age	STRING	NULLABLE
creation_date	TIMESTAMP	NULLABLE
last_access_date	TIMESTAMP	NULLABLE
location	STRING	NULLABLE
reputation	INTEGER	NULLABLE
up_votes	INTEGER	NULLABLE
down_votes	INTEGER	NULLABLE
views	INTEGER	NULLABLE
profile_image_url	STRING	NULLABLE
website_url	STRING	NULLABLE

votes

Schema | Details | Preview

Field name	Type	Mode
id	INTEGER	REQUIRED
creation_date	TIMESTAMP	NULLABLE
post_id	INTEGER	NULLABLE
vote_type_id	INTEGER	NULLABLE

Figure 3: Original tables in stack overflow dataset used for querying

More specifically, we used information about table schema[2] to first obtain the number of upvotes, downvotes and other types of votes (e.g. moderator review) by post id and stored these in a *post_vote_details* table. From there, we created a master table titled *all_data* that stitched together information on questions posted, users who wrote each post, and votes received into a single storage unit. This table included information on the content tags associated with each post in a single column (in a format such as 'python|pandas|dataframes'). Since we wanted to be able to look at simple trends in post

[2] Available at https://meta.stackexchange.com/questions/2677/database-schema-documentation-for-the-public-data-dump-and-sede

content over time, we further used SQL queries to parse out the tags column such that each tag for a specific post mapped to a different row. In addition, for topics that were of special interest, such as programming languages that could map to multiple tags (e.g. 'python' and 'python-2.7' are both tags about Python), we used string pattern matching to classify whether a tag was associated with the programming language. Similar approaches were used to classify whether posts were associated with any of the three major cloud computing systems, or with relevant big data tools and technologies such as Hadoop, Spark, Hive, Impala. The final two data tables produced are shown in Figure 4 and were used to visualize descriptive information on the Stack Overflow data using BigQuery's API in Tableau.

all_data

Schema | Details | Preview

Field name	Type	Mode
id	INTEGER	NULLABLE
title	STRING	NULLABLE
body	STRING	NULLABLE
answer_count	INTEGER	NULLABLE
view_count	INTEGER	NULLABLE
comment_count	INTEGER	NULLABLE
favorite_count	INTEGER	NULLABLE
score	INTEGER	NULLABLE
creation_date	TIMESTAMP	NULLABLE
owner_user_id	INTEGER	NULLABLE
tags	STRING	NULLABLE
age	STRING	NULLABLE
location	STRING	NULLABLE
reputation	INTEGER	NULLABLE
user_upvotes	INTEGER	NULLABLE
user_downvotes	INTEGER	NULLABLE
user_views	INTEGER	NULLABLE
upvotes	INTEGER	NULLABLE
downvotes	INTEGER	NULLABLE
othervotes	INTEGER	NULLABLE

tag_trends_full

Schema | Details | Preview

Field name	Type	Mode
Tag	STRING	NULLABLE
Month	INTEGER	NULLABLE
Year	INTEGER	NULLABLE
Num_Questions	INTEGER	NULLABLE
Num_Answers	INTEGER	NULLABLE
Num_Views	INTEGER	NULLABLE
Num_Comments	INTEGER	NULLABLE
Num_Favorites	INTEGER	NULLABLE
Total_UpVotes	INTEGER	NULLABLE
Total_DownVotes	INTEGER	NULLABLE
Total_OtherVotes	INTEGER	NULLABLE
Sum_Score	INTEGER	NULLABLE
Sum_User_Score	INTEGER	NULLABLE
Num_Users	INTEGER	NULLABLE
Lang_Type	STRING	NULLABLE
Cloud_Type	STRING	NULLABLE
BigData_Type	STRING	NULLABLE
BigData_Lang_Type	STRING	NULLABLE

Figure 4: Main derived tables used for descriptive analysis

Results

A basic overview of the data examined, along with some summary statistics in Figure 5. Overall, we looked at over 16 million questions posted over a 10-year period, with over 130,000 new questions posted each month. Most questions receive at least one answer, and on average, questions are associated with about 2150 views, 2 comments, 3 favorites and 3 content tags.

Data overview	
Number of questions	16845942
Number of unique users	3221661
Start date	2008-07-31
End date	2018-12-02

Summary Statistics	
Average questions / month	134768
Average answers / question	1.5
Average views / question	2154
Average comments / question	2.0
Average favorites / question	2.7
Average number of tags / question	2.97

Figure 5: Brief data overview and summary statistics

Looking at overall trends in stack overflow (see Figure 6), we can see that growth in the number of questions and unique users posting questions stabilized around 2013. However, there appears to be a reduction in the number of answers starting the same year, and a sharp decline in the number of views during this question. This suggests that community engagement from a responsive/content consumption perspective may be declining.

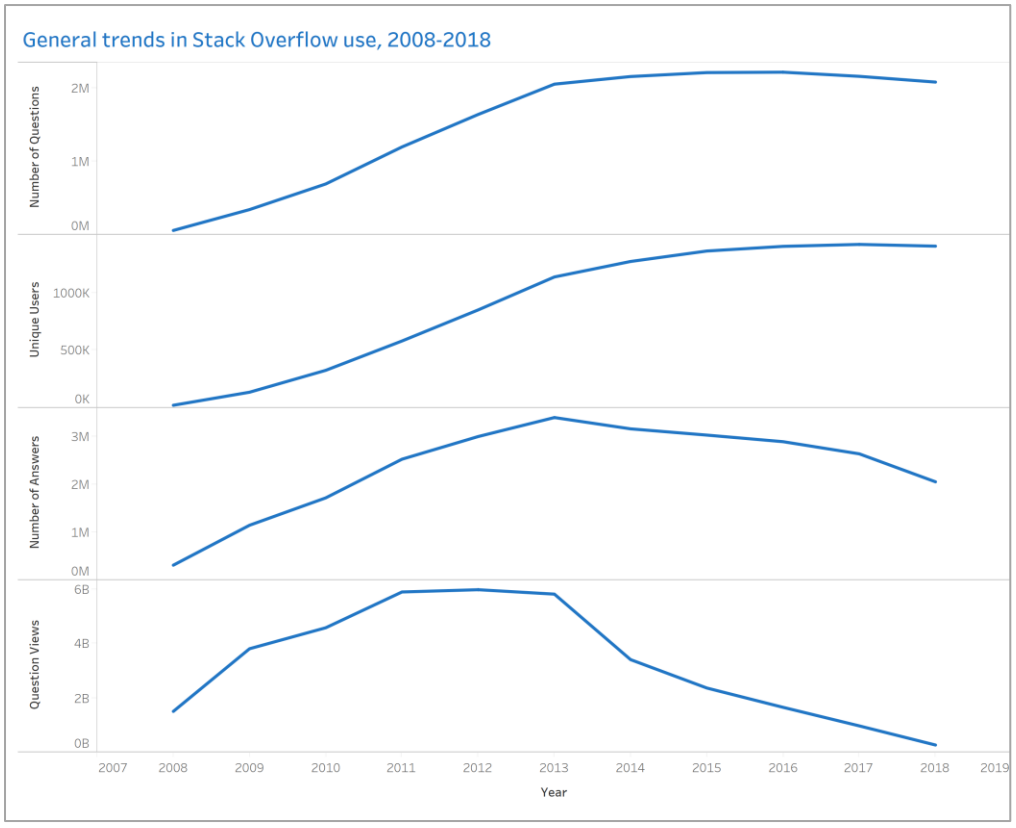


Figure 6: Trends in stack overflow data

Ultimately, we are interested in exploring patterns related to data science topics within Stack Overflow community. From Figure 7, we can see that Python and R, two languages primarily used for data analytics, are the main programming

languages that continue to experience growth even in 2018. Most other languages, including computer science languages such as C (capturing C#, C++), Java and website languages such as HTML started declining in the early to mid 2010s. After Python, JavaScript emerges as the language with the highest volume of questions posted.

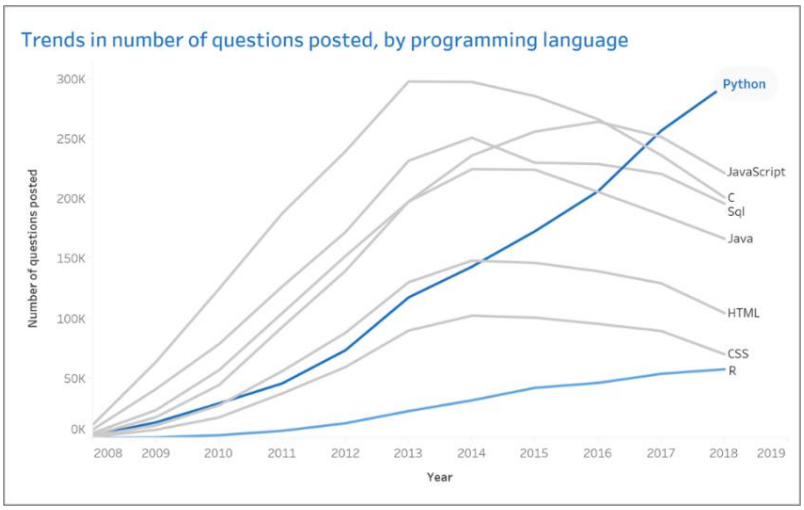


Figure 7: Trends in number of questions posted, by programming language

At the same time, Figure 8 shows that R may be associated with higher quality of questions, since it leads in the percentage of upvotes (as a share of total up and downvotes) and average question score in 2018, despite its relatively low question volume.

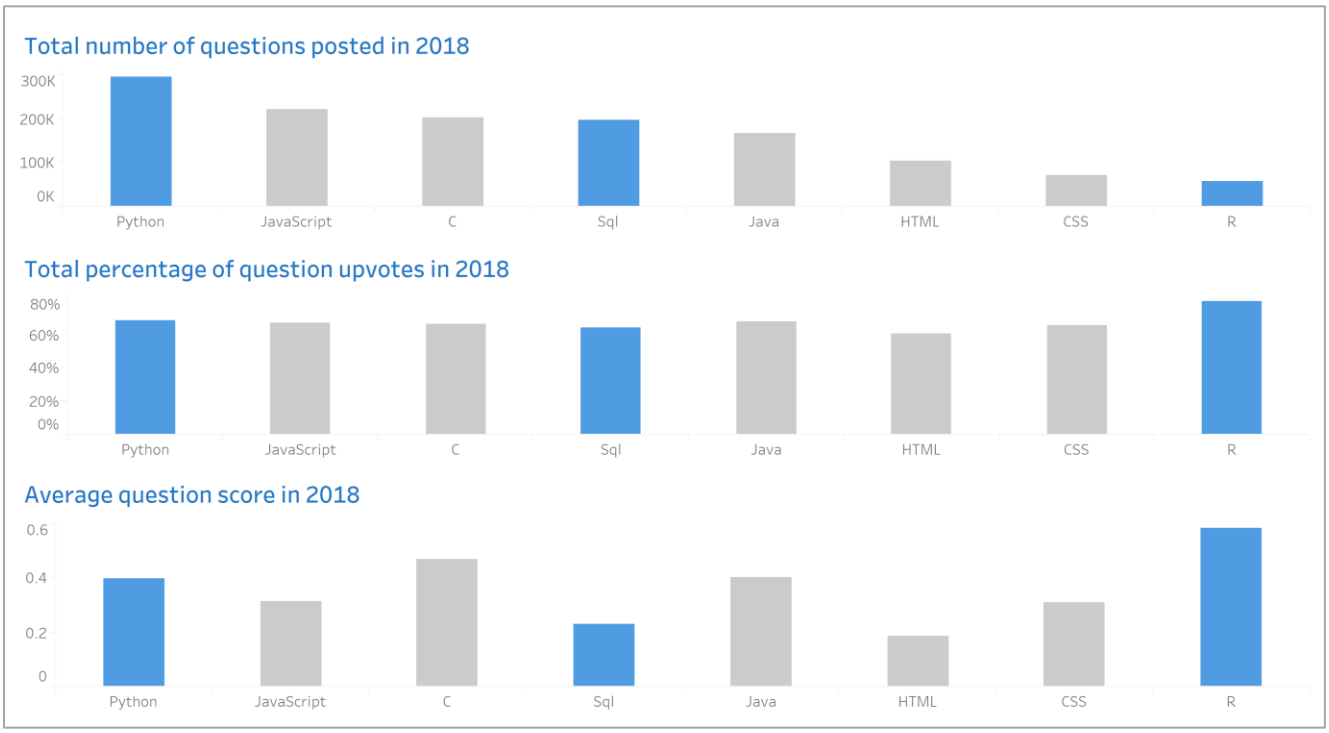


Figure 8: A closer look at question volume, upvote share, and score in 2018, by programming language

Looking at question trends and volume by cloud platform in figure 9, it is interesting to note that Microsoft Azure maintains a consistent lead in questions posted, when compared with the Google Cloud Platform and Amazon Web Services. Of course, it is worth noting that this data mainly represents the number of questions being asked on stack

overflow, which could be a function of both popularity and limited documentation/resources directly available. In terms of question scores, which capture the relative number of upvotes versus downvotes for a post, GCP emerges as the dominant ecosystem.

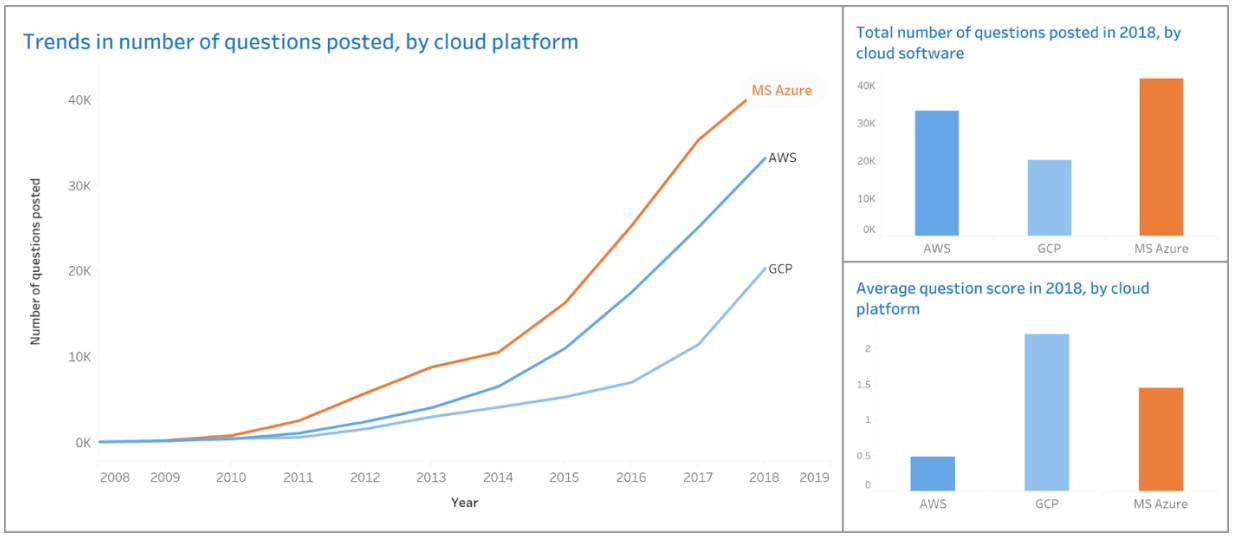


Figure 9: Questions posted by cloud computing platform

Lastly, we wanted to examine trends in searches related to big data tools and technologies. From the visuals in Figure 10, we can see rapid growth in questions about Spark beginning 2014, representative of the general growth in big data management in business during this period. Looking more specifically at big data languages, Scala appears to be associated with the most questions posted on Stack Overflow, but post volume is declining in recent years. By contrast, PySpark represents sharp growth in this category over the past few years.

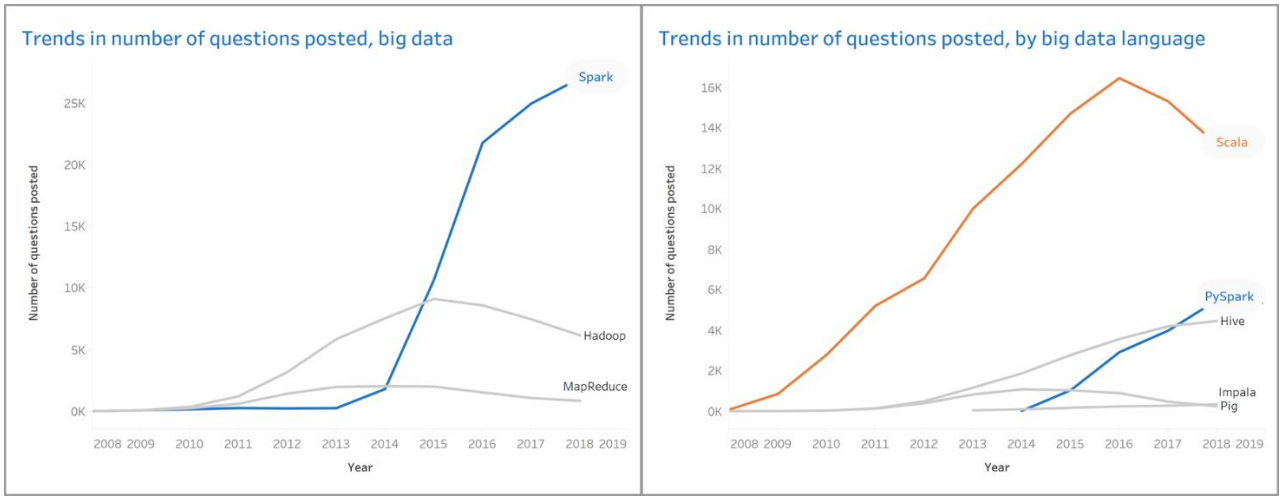


Figure 10: Questions posted about big data tools and technologies

Text mining for trending topics

Process

After descriptive analysis, we decided to conduct some text mining and build machine learning models to better understand trending topics on stack overflow. We chose to investigate Python and R because these two languages are what we use and learn the most and are among the most important toolkits for data scientists.

In order to accomplish our goals, we decided to do topic modelling with Latent Dirichlet Allocation (LDA) because LDA is arguably among the most successful recent learning algorithms for analyzing discrete data such as bags of words from a collection of text documents (Newman, Asuncion, Smyth & Welling, 2009). In our case, the dataset is large and distributed learning would be particularly useful.

Thus, we started our analysis by opening clusters on GCP Dataproc with one master and five worker nodes and connected to PySpark to utilize distributed computing power. We decided to work on data frames instead of RDDs, because its more efficient to work on data frames and Spark ML is better than Spark MLlib in high-level operations and machine learning tasks. To prepare for text mining, we used a custom data table created on BigQuery, as mentioned earlier. We opened pyspark and installed the jar package for BigQuery Connector. When setting up the connection, we had to designate a GCP bucket directory where we preferred to store the data. The data was then loaded on the Dataproc HDFS RDD and split into multiple files stored in the specified bucket directory. After loading data for the first time, for subsequent access, we could simply retrieve the data from our bucket without the need to set up the BigQuery Connector again.

In order to find recent popular topics on Stack Overflow, we selected posts from 2018 and combined post title and body to one text column. Only questions tagged as “python” or “r” were included. After selection, there were 242,708 posts for Python and 54,431 posts for R, and we did separate analysis for each.

We wanted to remove some of the common but non-distinguishing words for modeling convenience and accuracy, so we created a user defined function *cleanup_text* to filter out stop words. The function also removed special characters and filtered out any words with length less than two characters. Although the post bodies were stored in html, we managed to clean up all special formatting. In the end, the function returned a list of all the relevant words from the texts separated by commas.

To evaluate the importance of each word in the texts, we calculated the term frequency-inverse document frequency (TF-IDF). First, we applied the CountVectorizer function to convert the word list into sparse vectors storing information about the total number of different words across all texts, the words in this text and their corresponding occurrences in the same text. To increase efficiency for later analysis, we chose the top 10000 most frequent words for python and top 5000 frequent words for R and restricted to those appearing at least 2 times in different texts when vectorizing the words.

We chose CountVectorizer instead of Hashing-TF because of CountVectorizer’s reversibility of the vocabularies. It discards infrequent tokens to filter top features. We believe in our case, using CountVectorizer is more appropriate. The resulting sparse vectors have three components: the total number of different words among all posts (bound by text limit), a list of integers corresponding to the indexes of words, and a list of frequency counts in the order of word indexes.

Simply analyzing the text based on token counts can lead to biased results because some words might appear more frequent in general. For example, the word “the” is used so frequent regardless the context of the text; thus, emphasizing “the” when performing clustering is likely to generate less meaningful topic groups. The solution is to calculate the inverse document frequency, which reduces the importance of words that occur frequently across multiple texts and improve the weight of words that appear less often. Consequently, we can identify the keywords that distinguish one post from the other. By applying the IDF function, the sparse vectors replace the frequency counts with re-assigned weights.

Once the TF-IDF matrix was produced, we put the it into the LDA algorithm. The LDA model tries to identify the set of words that mostly likely constructed topics. Since the model involves an iterative process of placing words into different topics and assessing their probability of forming the topic, the process is very computationally expensive. Therefore, it is most appropriate to be conducted on a distributed system, especially with working with data at our scale. By setting the number of topics, the LDA model automatically generates groups of words based on their relevance to the topic, and we can interpret the topic themes intuitively based on our understanding of data science.

Results

After iterating on the above process for over 20 times and changing parameters such as the number of topics in models, we successfully identified 5 significant and distinctive topics for both Python and R. The top 10 words in each topic are shown in figures below.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Module installation	Dataframes, plots	Neural networks	Web development	Data types
install	column	tensorflow	django	list
pip	row	model	server	string
path	list	loss	user	value
command	dataframe	training	class	array
module	class	shape	form	function
running	data	input	message	number
files	text	return	password	score
version	color	data	return	name
run	plot	nan	test	data
installed	value	none	data	time

Figure 11: Topics identified for Python

For Python, it's clear that topics on data science shown in Figure 11 are trending: 3 out of the 5 topics identified are directly about data and modeling (topic 2, 3 and 5), while topic 4 is for web developing on Django: A Python-based free and open-source web framework. Not surprisingly, module installation is a significant question topic.

Some similar patterns were found for R in terms of topics on module installation and data frames, as shown in Figure 12. Shiny, the R package that provides a web framework for building web applications is also among the top topics. Data visualization in R is clearly a significant topic.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Module installation	Dataframes, plots	Neural networks	Web development	Data types
install	column	tensorflow	django	list
pip	row	model	server	string
path	list	loss	user	value
command	dataframe	training	class	array
module	class	shape	form	function
running	data	input	message	number
files	text	return	password	score
version	color	data	return	name
run	plot	nan	test	data
installed	value	none	data	time

Figure 12: Topics identified for R

In addition, we also tried to explore question overlap using the IDF vectors we already processed by computing cosine similarity between post titles. As an example, we took the sparse vector of two titles and calculated their cosine similarity, shown in Figure 12. A high similarity score means the two titles share a more similar topic.

Example of processed data

Attribute	Content
Title	How to use an array as an input of a dense layer
Words	[array, input, dense, layer]
Count vector	(1678,[44,50,769],[1.0,1.0,1.0])
IDF vector	(1678,[44,50,769],[4.58,3.89,5.83])

Cosine similarity used to assess overlap between pairs of questions

Example Question 1	Example Question 2	Overlap
How to convert bytes data to string without changing data using python	How to convert a string into a float when reading from a text file	0.27

Figure 12: Example of additional processing done for question overlap

Conclusion

Both the exploratory and machine learning analyses gave us interesting insight into popular topics on the Stack Overflow data science community. Our main takeaway from them is that data skills are trending. Posts on Stack Overflow, where professional programmers discuss their coding questions reflect what the computer science community is focusing on. What we have learnt in the program is very valuable in today's digital age, showcased by growth in questions about languages such as Python, R and big data frameworks such as Spark.

Moving forward, we believe there are opportunities to expand upon our approach with additional analyses, modeling and broader applications. For instance, while we only conducted our analysis on the questions, the dataset also has a rich set of information about the responses of each question and the users who posted them, including user reputation, upvotes, etc. This information can generate another set of interesting findings and should be explored future.

As an extension of the existing results, further modeling can be implemented using our analysis as the foundation. The cosine similarity of comparing question titles can be developed to process multiple questions quickly to identify similar questions, allowing recommendation of related posts in real time. Another potential modeling is to match tags automatically with questions to assist better search functions. Furthermore, combining text mining and probability estimation techniques, we might be able to predict high quality answers.

In addition, since the set of GCP tools is not specifically designed for text mining, it would be interesting to conduct analysis using similar tools on other data such as commercials or website data. It would also be helpful for us to learn about similar tools on Amazon Web Service or Microsoft Azure. Longer term, it would be beneficial to take some of the techniques implemented here, such as text mining of user-generated content, and apply them in a business setting to process customer reviews, and more.

Supplemental materials

Big data tools and technologies

- Summary:
  https://cloud.google.com/solutions/big-data/

- Cloud Composer (Manages Apache Airflow environments):
  https://cloud.google.com/composer/
  https://cloud.google.com/composer/docs/quickstart

- Google BigQuery
  https://cloud.google.com/bigquery/
  https://cloud.google.com/files/BigQueryTechnicalWP.pdf

- BigQuery ML
  https://cloud.google.com/bigquery/docs/bigqueryml-analyst-start
  https://cloud.google.com/bigquery/docs/bigqueryml-scientist-start
  https://cloud.google.com/bigquery/docs/bigqueryml-intro

- Example big data solutions
  https://cloud.google.com/solutions/building-real-time-inventory-systems-retail
  https://cloud.google.com/solutions/iot/
  https://cloud.google.com/solutions/architecture/optimized-large-scale-analytics-ingestion

- Use the Cloud Storage connector with Apache Spark
  https://cloud.google.com/dataproc/docs/tutorials/gcs-connector-spark-tutorial

- Use the BigQuery connector with Apache Spark
  https://cloud.google.com/dataproc/docs/tutorials/bigquery-connector-spark-example

Text mining on Spark

- Topic modeling with LDA
  Newman, D., Asuncion, A., Smyth, P., & Welling, M. (2009). Distributed algorithms for topic models. *Journal of Machine Learning Research*, *10*(Aug), 1801-1828.
  https://community.hortonworks.com/articles/84781/spark-text-analytics-uncovering-data-driven-topics.html
  https://medium.com/@connectwithghosh/topic-modelling-with-latent-dirichlet-allocation-lda-in-pyspark-2cb3ebd5678e