Stack Overflow Keyword Extraction System

CSC 591 002 - Group 9
Data Intensive Computing
Prof. Vincent Freeh
TA: Liang Dong, Fan Yang

Dated Sept-Dec'2018
Pavithra Iyer (piyer3)
Sanya Kathuria (skathur2)
Unnati Agrawal (uagrawa)

Abstract

We will discuss in this paper about an application that extracts and discovers the trend of keywords, languages or technologies over the desired time period using the Stack Overflow data. The keyword extraction is done using Spark. In the later sections we will see the whole pipeline for the process of extracting keywords and storing it in NoSQL database and how the application scales itself according to the user traffic. We will be able to associate big data technologies used to process data. We will see how spark is used in batch processing on static historical data.

1 Introduction

With the rapidly increasing volumes and variety of Stack overflow data, there is a need to build a system which can not only store the historical data to retrieve trends for keywords or technologies over any desired time frame but also maintain the real time incoming streaming data. In this paper, our focus is to to develop a scalable web application that extracts and discovers the trend of keywords, languages or technologies over the desired time period using the Stack Overflow data. Apart from dealing with challenges of big-data like data collection, transformation and keyword extraction, we also want to store our data in a NoSQL database and build an application which scales with user traffic.

Stack overflow is a much loved programmer question and answer site written by two guys nobody has ever heard of before. Well, not exactly. It was created by top programmer and blog stars Jeff Atwood and Joel Spolsky.It is noticed that every 30 minutes, 1/3 rd of the programmers use Stack overflow. The static data analysis for Stack overflow data becomes a big data problem as scaling up is becoming in feasible and the number of data sources are exploding. Hence, we aim to deal with high volume and variety of data in our solution for this Big data problem.

With this project we intend to extract the keyword from Stack Overflow data. But first there should be a good understanding of what is the keywords with respect to the problem statement. When people post various questions on Stack Overflow, they are generally associated with a technology. It could be a question related to technologies like "JAVA", "C++", "C" etc. So in this project when a user inputs a time frame in the interface, the result would be the most common keyword for that particular time frame. This would help in understanding the technology trend of computer science.

1.1 Motivation

We strongly believe that having a good understanding of the big data technologies is crucial for any computer scientist as every software company today collects large amounts of data and analyzes it to extract value in business out of it. We took this course to learn about the different big data technologies and work on a project as part of this course which at least touch bases on most of the big data technologies.

All three of us wanted to deep dive into learning Spark and run batch jobs in a cluster. Cassandra was the obvious choice for a distributed database. Also, we desire our web application to scale with the user traffic. AWS Lambda is great option for small functions and takes care of auto-scaling.

Submitted for CSC 591-002 Data Intensive Computing Course.

2 Architecture

Since we are dealing with huge volumes of data, building a distributed data processing framework,was the main task. The system should be scalable, have low latency, fast in memory processing, store the results, report the results real time on REST API and serve ad-hoc queries. The keyword extraction should be as fast as the time taken to post a question. For our problem statement we have dealt with the static historical data. We need to store the historical data as well as the results as people might want to go back to the results.

The technology stack used are Spark cluster for calculating the word count, Cassandra for storing out database, lambda function which is available in Amazon Web Services. Spark cluster and Cassandra are deployed on AWS EC2 instances running on the same VPC. We need to ensure they are the instances are in the same VPC so that these components can communicate with one another. Also, the lambda function has to be given EC2 access privilege and added into the same VPC as the EC2 instances. Ensuring the networking is right is crucial in any distributed systems projects. Also, an API gateway with POST method needs to configured to send the POST request issued by the front-end (to determine popular keywords) to the lambda function.

The architecture of the system requires major components:

1. Data Collection where data is collected from Kaggle and cleaned using Python scripts.

2. Data Transformation and Keyword extraction using Spark for fast and parallel computations on our data.

3. Data Loading in a distributed database using Cassandra.

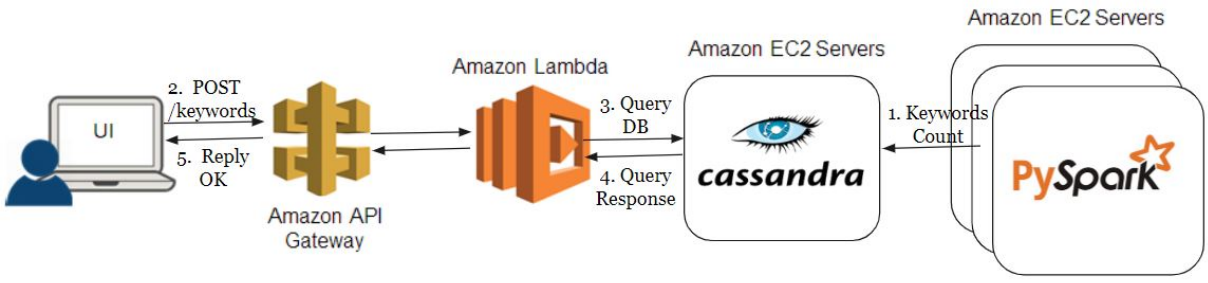
4. Infrastructure on AWS. WE deploy scalable serverless computing for backend microservice using AWS Lambda and cloud instance using AWS EC2.

2.1 Components

The data is used for the problem statement is collected from kaggle.[15]. For data cleaning purpose, we wrote a script in python. Our whole infrastructure was based in AWS where we have used EC2 for cloud infrastructure, AWS Lambda function for scalable server-less computing for back-end micro service. To have fast and parallel computations, Apache Spark was used and Cassandra for distributed database. In the later sub sections we have discussed which these choices were made to carry on the task and not their alternatives.

2.2 Main flow of architecture

As from the figure below we can see the individual components and how they are tied together.



2.2.1 Data pre-processing

From the dataset[15], we can see that there are two files, the answers and questions file.One file consists of the answers written by the users and the other which consists of the questions asked by the user. Since our task was calculating the keywords, both the files were relevant for our problem

statement. It is easier if we process the data which is present in a single place. So, both the files were merged into a single place. We extracted the timestamp and the Content columns from the data. Content was cleaned to remove the html tags and special characters. Also, our data set consisted of the "Tags" which were the keywords which we had to count. Hence, we removed that column from our dataset as that is what we wanted to predict. This is how dataset was ready to be fed into spark clusters. The figure below shows the data format which we got after the it was cleaned.

	CreationDate	text
0	2008-08-01T14:45:37Z	p a href http svnbook red bean com version control with subversion a p p a very good resource for source control in general not really tor
1	2008-08-01T16:09:47Z	p i wound up using this it is a kind of a hack but it actually works pretty well the only thing is you have to be very careful with your semico
2	2008-08-01T19:36:46Z	p i ve read somewhere the human eye can t distinguish between less than values apart so this is something to keep in mind the following
3	2008-08-01T23:49:57Z	p yes i thought about that but i soon figured out that another domain specific language dsl would be a bit too much p p essentially they r
4	2008-08-02T01:49:46Z	p a href http www codeproject com articles c script the missing puzzle piece oleg shilo s c script solution at the code project a really is a g
5	2008-08-02T03:00:24Z	p i would be a bit reluctant to use nested classes here what if you created an abstract base class for a multimedia driver to handle the bac
6	2008-08-02T04:18:15Z	p you might be able to use ironruby for that p p otherwise i d suggest you have a directory where you place precompiled assemblies then
7	2008-08-02T06:16:23Z	p you could use any of the dlr languages which provide a way to really easily host your own scripting platform however you don t have to
8	2008-08-02T15:33:13Z	p no what you re doing is fine don t let those people confuse you p p if you ve written the web services with net then the reference proxie
9	2008-08-02T18:16:07Z	p isn t it also a factor which order you set up the colors p p like if you use dillie os idea you need to mix the colors as much as possible is f
10	2008-08-02T19:03:52Z	p my first thought on this is how generate n vectors in a space that maximize distance from each other you can see that the rgb or any oth
11	2008-08-02T23:40:04Z	p for my projects i alternate between sql compare from red gate and the database publishing wizard from microsoft which you can downl
12	2008-08-02T23:51:09Z	p i ve taken to hand coding all of my ddl creates alter delete statements adding them to my sln as text files and using normal versioning u
13	2008-08-03T00:22:03Z	p if you have a company buying it toad from quest software has this kind of management functionality built in it s basically a two click ope
14	2008-08-03T00:37:03Z	p i work the same way karl does by keeping all of my sql scripts for creating and altering tables in a text file that i keep in source control ir
15	2008-08-03T01:38:02Z	p i agree that scripting everything is the best way to go and is what i advocate at work you should script everything from db and object cr
16	2008-08-03T11:41:38Z	p with the built in stuff you can t as using or will replace the revision and build numbers with a coded date timestamp which is usually alsc
17	2008-08-03T18:46:33Z	p what source control system are you using p p almost all of them have some form of id tag that gets expanded when the file is checked i
18	2008-08-03T20:45:27Z	p i m partway to my solution with this entry on msdn don t know how i couldn t find it before p p user machine hive br subkeys and value
19	2008-08-03T20:48:47Z	p first yes this is something that belongs in the application for the exact reson you specified what happens after new user profiles are crea
20	2008-08-03T21:17:33Z	p i m guessing that because you want to set it for all users that you re on some kind of shared computer which is probably running under
21	2008-08-03T22:34:06Z	p despite what the a href http msdn microsoft com en us library x kd c vs aspx rel nofollow msdn article a says about user machine hive it
22	2008-08-04T01:07:14Z	p very roughly and from memory since i don t have code on this laptop p pre code using oledbconnection conn new oledbconnection cor

As we can see in the picture, we have one column as the CreationDate which tells us when the text was posted and the text tells us the content posted by the user which could be a question asked or and answer written for a question to a particular timestamp.

2.2.2 Keyword Extraction

This part takes place in the Amazon EC2 servers with the help of spark. Spark should be first installed and configured in EC2 [5]. Spark is available in EMR servers in AWS as well, which need no configuration and installation. One could even use that but we wanted to increase the complexity of the whole project. AWS educate could not be used for our setup because it has a few restrictions. It does not allow the user to define IAM roles to add permissions. EC2 is one of the resources which the Lambda needs and this cannot be configured with AWS Educate account. We decided to do our setup in AWS free tier account which does not have any restrictions and gives 750hrs free per month for EC2 servers and 1 million function calls for AWS Lambda function.

Once the data is present in EC2, we submit a job to spark. The primary step in spark job is to parallelize the data to create RDD[8]. The tasks are executed in sequence of multiple maps and reduce. In the map phase, keyword count is done for a document. For a time frame, a popular keyword is counted. Reduce task, assembles it. Multiple documents published on a certain date could have occurrences for the particular keyword in the document so it assembles it. Finally, when the keyword count is found, we write it into Cassandra database.

In the map phase, we had two approaches in mind. Our data-set had lot of stop words like "a", "the", "of" etc which were irrelevant for our results and their count would be more than the keywords. One way was to come up with all the stop words, and ignore them in our map phase. But building a exhaustive list of such words would be difficult. So instead of that we approached the second way

where we created a dictionary and included all the relevant keywords in the dictionary. Finally, in the map phase, the count for keywords present in that dictionary was calculated.

2.2.3 Loading Data into database

Cassandra database is installed in EC2 servers as well. One could install Cassandra locally in Spark nodes as well. We have decide to install it on separate instance because Spark is a heavy application and since we were using micro instances, we wanted it to be decoupled. Cassandra has to be accessed from remote hosts. In order to do that we had to modify few configuration parameters in the cassandra.yaml configuration file. Once the keyword count is found, we push it to Cassandra database.

timestamp	keyword	count
2008-08-21	java	4
2008-08-21	javascript	8
2008-08-21	jquery	1
2008-08-21	perl	1
2008-08-21	python	1
2008-08-21	sql	6
2008-08-03	c	3
2008-08-03	sql	2
2008-08-13	c	9
2008-08-13	html	9
2008-08-13	java	5

3 Static Analysis Data Pipeline

We have built a front-end for finding the popular keywords in user input time frame. We used HTML, CSS and Javascript to create and design the front-end. An HTTP request is sent to AWS API gateway with POST method. The form data is added to the body section of the HTTP request. The API Gateway invokes the lambda "popularkeywords" function with HTTP request as an event object. Lambda connects to the database hosted on EC2 instance and queries for the popular keywords. It sends back the results to the client as a HTTP response and adds the answers in the body section of the response as a JSON object. The front-end plots a graph using the Google visualization API so that it becomes easy for the user to visualize and analyze the data. AWS Lambda would ensure that the back-end will not break if the demand for our application increases. This could be validated by using the ApacheBench tool which can initiate multiple simultaneous requests per second.

4 Why this technology stack?

This section provides a brief explanation of why we chose the different components to build our system.

4.1 Spark : Distributed Computation

Spark is greatly used for distributed data processing. It is used for query processing, streaming data, transforming and analyzing data at a large scale. Spark has a rich set of APIs. It has great speed and there are claims that Spark can be 100 times faster than map reduce [9]. RDDs are persisted in memory which provides in greater speed typically 10 times faster. Such processes which require iterative approach, spark works better in such cases as RDDs allow map operations to happen in memory and they need not be written to disks which generally is slower [10].

4.2 Cassandra : Distributed NoSQL Database

As we wanted to store, word count of a keyword which was associated with a particular time, NoSQL database was used. For our problem statement, a dataset was required which could be globally distributed and was fault tolerant. The keyword count is stored in the database and if at some point the database fails, then the whole application would go down [11]. Also, with time, in this application, we would get more data and more keywords would be generated and more data would need to be stored. Cassandra helps in this as it is linearly scalable. Cassandra is easy to query,[11] which helped in fetching data from it according to the user provided timestamps.[12]

4.3 Amazon lambda : Serverless Function

AWS Lambda is becoming popular as it enables serverless computing. It has several other advantages like auto-scaling, zero infrastructure maintenance and the customers need to pay only for the number of function calls rather than the server up time. If our web application becomes popular, then AWS Lambda will take care of scaling the back-end for millions of users.

5 Github Link

The github link for our project is `https://github.ncsu.edu/uagrawa/DIC\textunderscorePROJECT.git` .

6 Conclusion

1. Data Preprocessing and Configuration set up took 80% of the time and effort.
2. Spark cluster when configured properly made the system fault - tolerant.
3. Additionally, Spark cluster took hardly half an hour to extract top keywords from our dataset.
4. Overall, Distributed Computing was way faster than using single machine for large datasets.

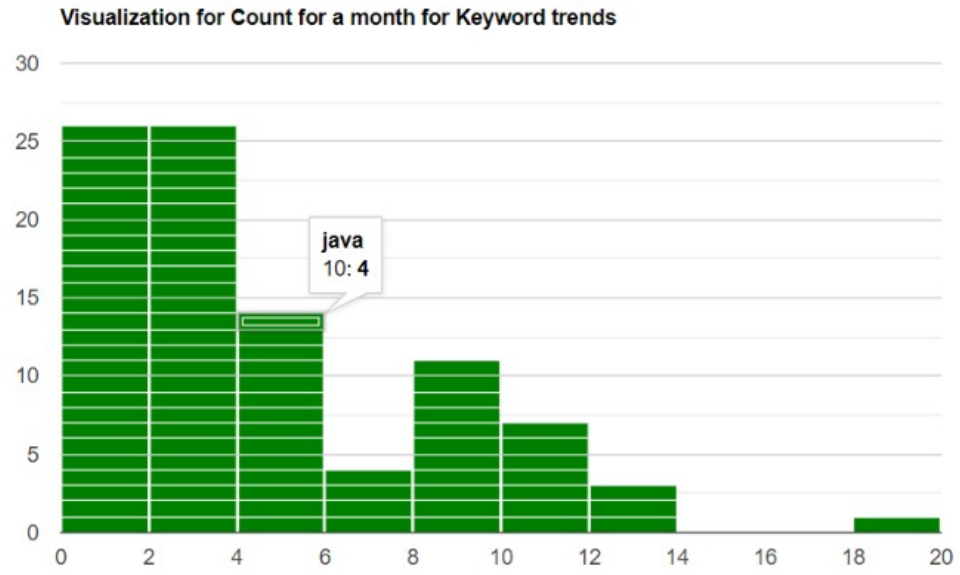
6.1 Features of our System

Although this problem focuses on static data analysis. Still, building a distributed data processing framework can be a complex task for non-streaming data as well. Below are the features our system caters to:

1. Low Latency
2. Scalable
3. Faster in-memory Processing
4. Storing the results
5. Reporting the results real-time on REST API
6. Ad-hoc Queries.

6.2 Results

Here's a histogram of Stack Overflow Keyword trends in Various Technologies:



The above graph shows the visualization for keywords (Java here) being displayed on the 10th day with count 4.

6.3 Dependencies

1. Dataset of Stack Overflow was not available with high volume. So we had to replicate it for our problem statement.
2. Spark for processing of data.
3. Cassandra for storing scalable and highly available data on the system.
4. Setting up AWS Lambda to interact with EC2 servers. We added IAM role with EC2 access. We had to configure Lambda and all EC2 instances in same VPC. AWS Educate does not give you permissions to edit IAM roles.
5. Python 2.7 preferably.

6.4 Issues Faced

1. Since the back-end REST APIs are written in Lambda, DynamoDB would have been easiest to configure with it. However, we wish to experiment with installing and configuring Cassandra on EC2 servers as it also offers scalability and high availability. We anticipated some issues in setting up the Lambda functions to interact with Cassandra.
2. AWS Educate does not give user permissions to create IAM roles
3. Lambda has to be configured in same VPC as AWS instances and the security credentials should be configured properly.
4. Efficiently use the AWS resources to finish our project goals.
5. Splitting the data into training and test data and convert into RDD.
6. Initially we tried to configure the system in VCL(virtual computing lab), but it did not work as it had lot of issues with the IP tables and they would need to be configured.

6.5 Future work

1. We could extend our system to support stream data from Stackoverflow. One could read data from Stack Overflow API and do a keyword extraction on a real time data. This could be done by using spark for streaming data instead of batch processing.
2. It could be tested to handle large data by adding more nodes to Spark Cluster. As the time passes by, more data would be collected. This system could be scaled accordingly by adding more nodes and computations would be done simultaneously. In this way, efficient scaling could be done without using the system being slow.
3. We could make it flexible by extracting keywords from technical blogs like Medium, Hacker News, Hacker Noon, etc. The scope of the project could be expanded by not just limiting the user posted content as data from Stack Overflow but also extracting it from various other sources in the internet.
4. More visualization can be added for analyzing Stack Overflow data. To make the experience better for user one could add more graphs.

References

[1] http://spark.apache.org/

[2] https://www.tutorialspoint.com/cassandra/

[3] https://stackoverflow.com/questions/36133127/how-to-configure-cassandra-for-remote-connection

[4] https://www.inertia7.com/projects/36

[5]https://medium.com/ymedialabs-innovation/apache-spark-on-a-multi-node-cluster-b75967c8cb2b

[6]https://haydenjames.io/when-to-use-amazon-ec2-t1-micro-instances/

[7]https://stackoverflow.com/questions/26659647/mapreduce-or-spark-for-batch-processing-on-hadoop

[8]https://jaceklaskowski.gitbooks.io/mastering-apache-spark/spark-rdd.html

[9]https://mapr.com/blog/spark-101-what-it-what-it-does-and-why-it-matters/

[10]https://www.edureka.co/blog/apache-spark-vs-hadoop-mapreduce

[11]https://blog.pythian.com/cassandra-use-cases/

[12]https://www.datastax.com/nosql-databases/benchmarks-cassandra-vs-mongodb-vs-hbase

[13]https://spark-packages.org/package/anguenot/pyspark-cassandra

[14]https://www.rosehosting.com/blog/how-to-install-apache-cassandra-on-ubuntu-16-04/

[15]https://www.kaggle.com/stackoverflow/stacksample