University of New Haven

Geo-location Clustering using the k-means Algorithm

BY YASASWINI SURYADEVARA

UNDER THE ESTEEMED GUIDANCE OF PROFESSOR VAHID BEHZADAN

DISTRIBUTED AND SCALABLE DATA ENGINEERING

DSCI-6007

Motivation

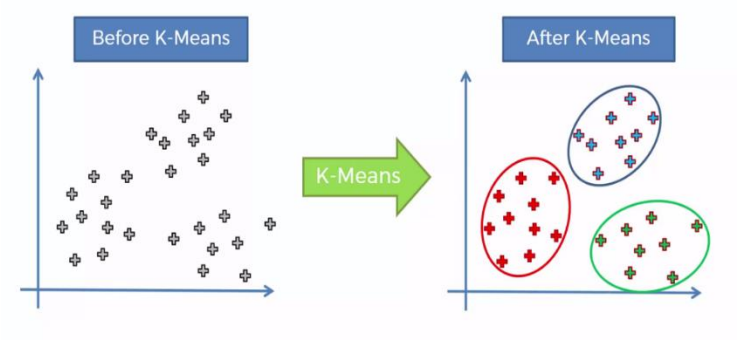
The goal of the project is to use SPARK to implement an iterative algorithm that solves the clustering problem in an efficient distributed fashion. In our case, we are implementing a geo-location clustering from dataset with latitude-longitude pairs. The algorithm we are using is k-means clustering algorithms that groups a given dataset into k clusters based on similarity measures like Euclidean distance. The results will be displayed on the map of the world to visualize the distribution of the clusters.

In this project we learn more about how to use Apache Spark, it is an open source, general-purpose distributed computing engine used for processing and analyzing a large amount of data. Data engineers are responsible for these higher-level operation at massive scale:

- data ingestion;
- data transformation or wrangling;
- data storage; and
- data access.

In data engineer perspective with respect to Spark we will use Structured Streaming to ingest real-time data from either streaming source such as Kafka or Kinesis or S3 or HDFS we may ingest raw data from secondary sources such as RDBMS, S3/HDFS, data warehouses, or other data repository.

Clustering algorithms can be used to determine which geographical areas are commonly visited and "checked into" by a given user and which areas are not. Such geographical analyses enable a wide range of services, from location-based recommenders to advanced security systems, and in general provide a more personalized user experience.



Documentation of Approach:

Clustering data is actually a very complex topic. There are various methods and algorithms available that fit different needs. The best solution for clustering points is using the Euclidean distance as dimension. One example of an algorithm for clustering geo points is k-means clustering. It is a popular approach in data mining, this algorithm clusters a number of observations into a given number of clusters. Each observation in this cluster belongs to the cluster with the nearest mean. From a complexity theory view point this is an NP-hard problem.

Amazon Services used: EC2, EMR, S3

Application: Spark

Coded in Pyspark using Jupyter Notebook

For this project we are going to use apache spark to perform geo location clustering k-means clustering. Took a glance on k-means clustering which we are already familiar with and revised some spark modules as we will be using pyspark for most of the processing and visualizations. First, I am going to create a cluster by following the system configuration I mentioned below. Using EMR cluster created a cluster and SSH into cluster and connected to terminal then opened jupyter on port 8888. In the jupyter notebook imported all the necessary imports after importing uploaded the files of device data to S3 using pyspark and analyzed the files then cleaned the dataset by performing necessary steps using map and filter functions and then created a created a clean data frame and again uploaded the clean dataset to S3, then plotted latitude and longitude in the map. After doing the same thing to synthetic location data, uploaded latitude and longitude to the S3 bucket from the S3 bucket read the files then visualized the synthetic data then did the same thing to DBpedia data. Next finding the k-means using closestpoint, add points, Euclidean distance, great circle distance. I am going to use both Euclidean distance and great-circle distance with different k values and check the results for k-means Euclidean distance Algorithm and great-circle distance k-means algorithm ,find which is more accurate and then put all the data into the csv files and make visualizations

System Configuration

Steps followed to set up the EMR Cluster

Step1: Open your AWS Account. Go to your AWS Console

Step2: Go to services, search for EC2 and click on it

Step3: Click on Launce instance

Step4: Select Ubuntu Server 18.04 LTS – 64-bit(x86)

Step5: Select General Purpose t2.micro type and click on Next configure Instance Details

Step6: Click on Add Storage and put size (15Gib-30Gib)

Step7: Click to add name tag and give a name if you wish

Step8: Click on Configure security group and select create new security group, you can give it a name if you want.

Step9: Click on Review and Launch

Step10: Click on Launch, A dialogue box will appear click on create new key pair and give any name you want. Download the key pair.

Step11: Click on launch and wait for the instance to launch (Make sure all security checks run)

Step12: Now go to EMR cluster in amazon services

Step13: Click on create cluster

Step 14: Give a cluster name

- Put launch mode as cluster
- Application must be spark
- Choose instance type you want I chose m4.xlarge
- Choose the ec2 key pair that is created while launching EC2 instance

Step 15: Click on create cluster and wait for the cluster to be ready

Step16: Go to master security group of created cluster and click on add rules in the inbound rules and add two security groups

1.SSH port at anywhere

2.Custom TCP with port 8888 anywhere

Now connect to your jupyter notebook with port 8888 in the terminal

Big data application/dataset:

Review k-means algorithm

k-means is a distance-based method that iteratively updates the location of k cluster centroids until convergence. The main user-defined ingredients of the k- means algorithm are the distance function (often Euclidean distance – for geo-location data we will also explore the great-circle distance) and the number of clusters k. This parameter needs to be set according to the application or problem domain. Kmeans algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid.

Data preparation steps:

Processed the device status data into a clean and standard form by doing the following steps:

Understanding the data set and uploading the data/devicestatus.txt into S3 bucket

Using Pyspark in jupyter notebooks to load the dataset

Separating the, for this I used .map() function to take each column value separately to achieve a cleaner dataset

Filtering out the records that do not parse correctly, I used .filter() function to achieve this

Extracting the necessary fields date, model, device ID, and latitude and longitude using .map() function

Filter out locations that have a latitude and longitude of 0

Split model field by spaces to separate the manufacturer from the model, for this I used .map() function

Save the extracted data to comma delimited text files in S3

All the necessary data preparation steps are done the data is presented cleanly

Confirm that the data in the file(s) was saved correctly.

I took a screenshot of the comma separated text files

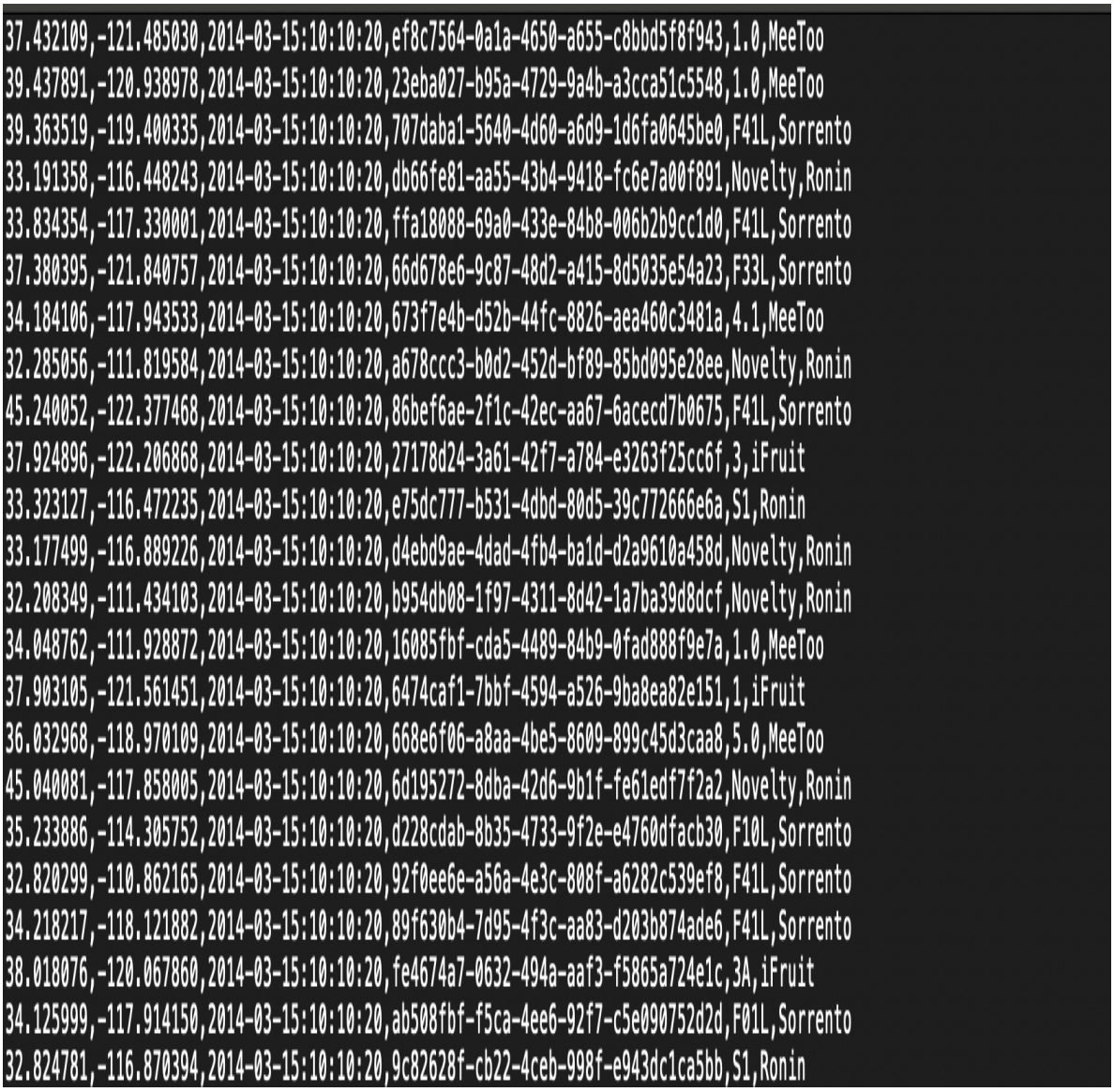


Image: Screenshot of comma delimited text files in pre-processed data set

Visualize the (latitude, longitude) pairs of the device location data.

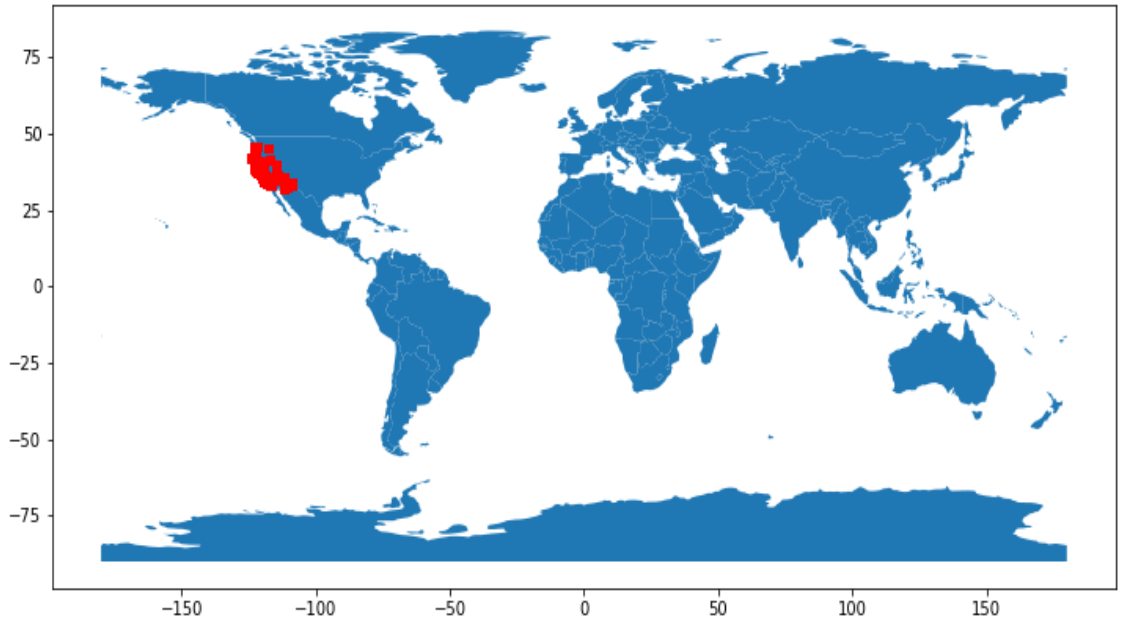


Image: Visualizing the device location data

Get and Visualize synthetic location data

Uploaded the synthetic clustering data sample_geo.txt to S3

Visualized the (latitude, longitude) pairs.

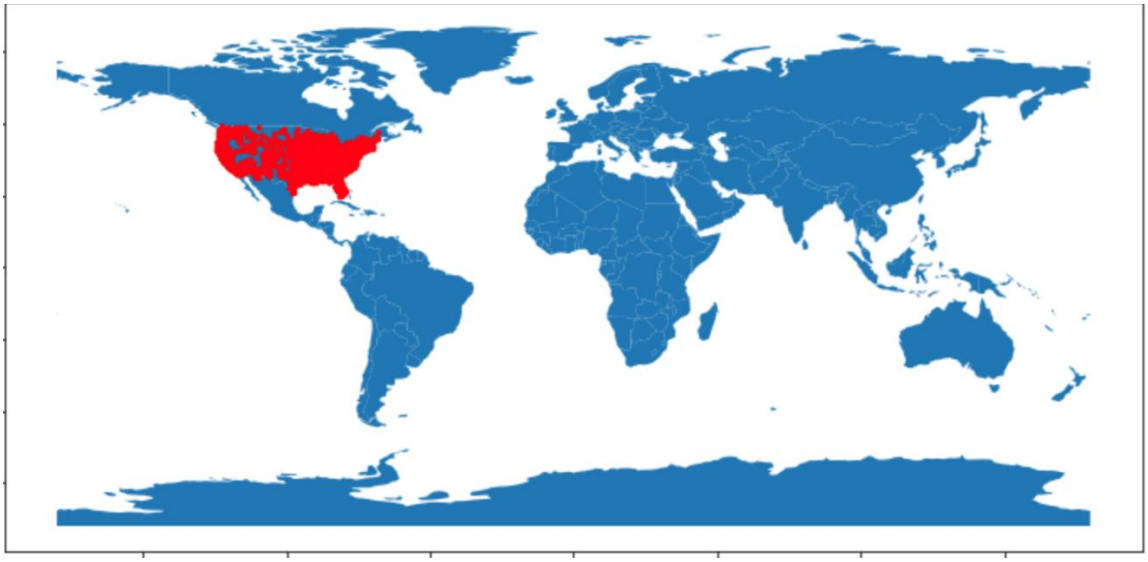


Image: Visualizing synthetic location data

Step 3: Get and Pre-process the DBpedia location data

Uploaded the large-scale clustering data of (latitude, longitude) pairs extracted from DBpedia to S3

Each record represents a location/place that has a Wikipedia article and latitude/longitude information. The format is: lat long name_of_page. In total, there are 450,151 points in a 2D space

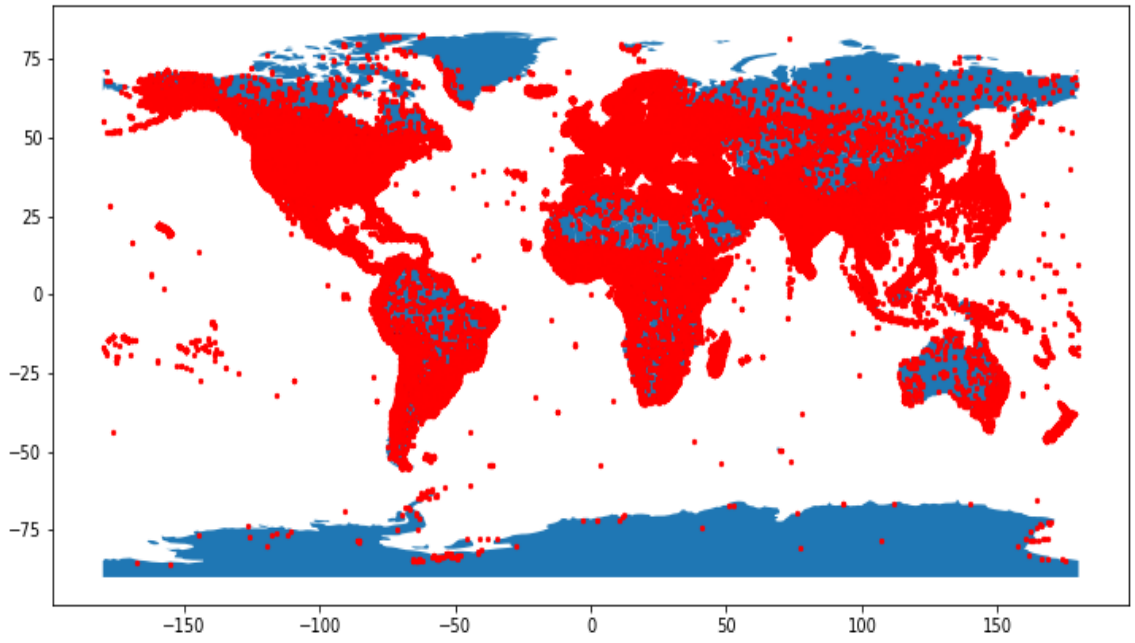


Image: Visualizing the dbpedia location data

Problem 3: Clustering Big Data – k-means in Spark

Step 1: Understanding Parallel Data Processing and Persisting RDDs

Took a quick review of Resilient Distribute Datasets (RDDs) to persist an RDD (at least once) in your k-means implementation and made myself familiar with using the Spark Application U

Step 2: Implementing k-means

I have used

• closestPoint: given a (latitude/longitude) point and an array of current center points, returns the index in the array of the center closest to the given point

• addPoints: given two points, return a point which is the sum of the two points.

• EuclideanDistance: given two points, returns the Euclidean distance of the two.

• GreatCircleDistance: given two points, returns the great circle distance of the two

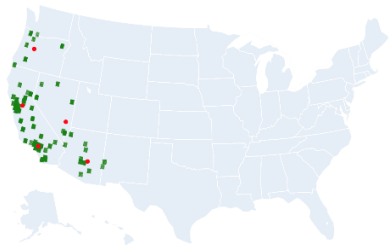
Noted, that the addPoints function will be used to compute the new cluster centers. The used distance measure (Euclidean or great circle), as well as the parameter k (number of clusters) are read as an input from the command line. Selected suitable convergence criterion and created a variable convergeDist that will be used to decide when the k-means calculation is done, i.e. when the amount the locations of the means change between iterations is less than convergeDist.

After defining the major functions used spark sc to read the data and tooksome random samples from the points as original centers. Then used map to group the points according to their closest centers and counted each point. Then used reduce by key to sum up the points with the same index of centers as well as their counts and divided the sum by total countto find out the new centers for each cluster when ConvergeDist reached certain value loaded output in csv files and visualized the data.

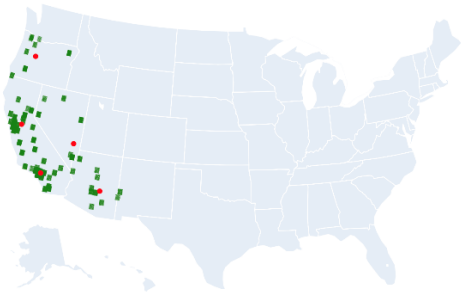
Visualization of Device location data K-means clustering

Visualization of Euclidean distance of device location data with k=5

Plotted Euclidean distance and Great circle distance for k=5 is the same for device location data .The red dots represent the centroids.



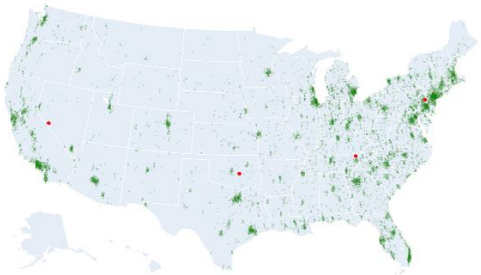
Visualization of Great circle distance k-means clustering of device location data with k=5



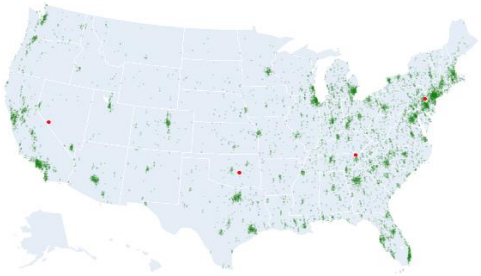
Both Euclidean distance and great circle distance k-means clustering gave the same results for k=4 for synthetic location data

Visualization of Synthetic location data K-means clustering

Visualization of Euclidean distance of synthetic location k-means clustering data with k=4

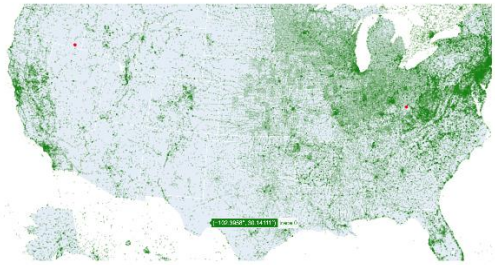


Visualization of Great distance of synthetic location data k-means clustering data with k=4

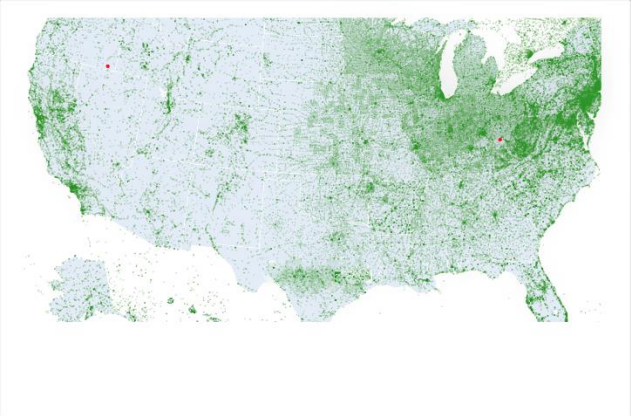


Visualization of DBpedia location data K-means clustering

1)Visualization of Euclidean distance of DBpedia k-means clustering data for k=2



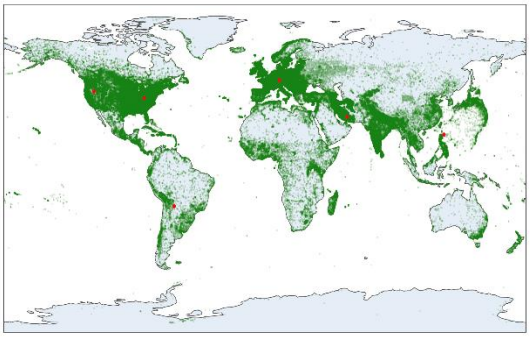
Visualization of Great Circle Distance k-means clustering of DBpedia data for k=2



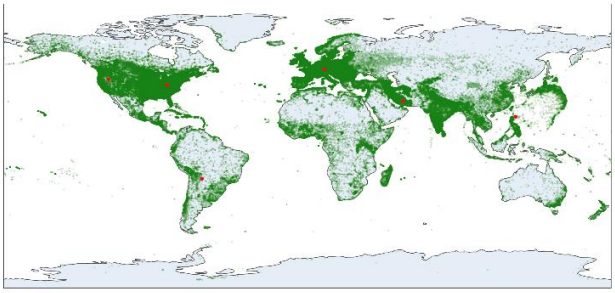
Observation: For k=2 both Euclidean distance and great circle distance gave similar outputs, the centroids are represented by red dots. The centers separate east and west closer to the area having relatively high density. Apparently, the cluster at west side has more geo points and has high density.

2)Visualization of Euclidean distance and Great Circle distance k-means clustering of DBpedia data for k= 6

Euclidean

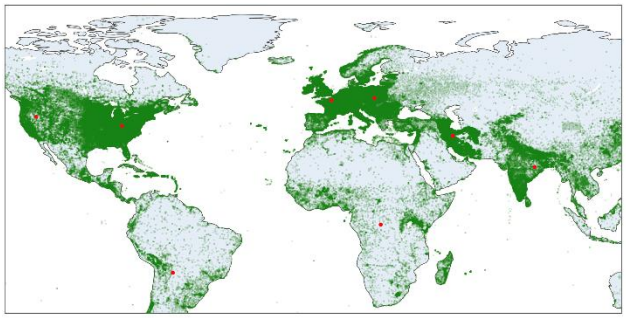


Great-Circle

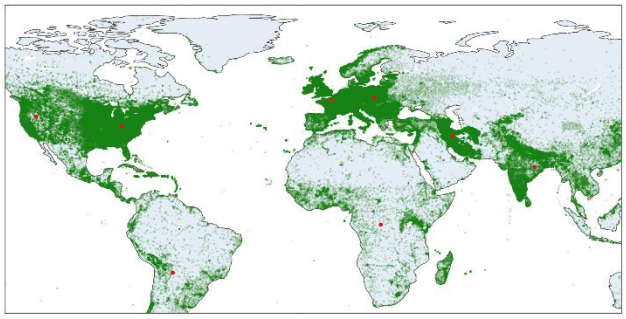


Observation: For k=6 both Euclidean distance and great circle distance gave similar outputs, the centroids are represented by red dots. The centers separate the continents ,as there are lot of geo points in North America it consists of two centers due to high density and two cluster centers in asia , one in Europe and one in south America. This is not bad, but not accurate.

Visualization of Euclidean distance and Great Circle distance of DBpedia data for k= 8



Euclidean distance



Great-Circle Distance

Observation: For k=8 we can see that both Euclidean and Great circle distance k-means algorithm gave similar distribution of k clusters. The centroids are represented with red dot. We can observe that the distribution is slightly better at k=8 as the k value increases the accuracy increased until now. Still North America contains two centroids and South America has one. Now Europe has two centers two in Asia and one in Africa, which makes it better than before. So, it seems that as the k-value increases from 6 to 8 the clustering distribution is more accurate. But as the k further increases to 10 the data might get weird because there might be more data points in Europe due to high density at one point which is not good.

Analysis:

1.Using distance as the only metric to clustering without considering the border lead to the division of same country or continent in cluster result

2. The results of k-means geo location based on Euclidean and great circle distance between two points shows the degree of aggregation from the aspect of geo location. Closer the points are more similar they are, it is more probable that they are the same cluster.

3.As we use distance as measurement clusters always close to density centers which is obvious because its average summation of data points

4.The cluster is not bad when k=8, it is better than k=6.k means algorithm does have many disadvantages but it is the simplest model. Since the result is distance based the cluster will be similar to circular shape. The k means algorithm may not be robust but does perform well for uniformly distributed data.

Runtime Analysis:

	Device Location Data	Synthetic Location Data	DBpedia Location Data
With persistent RDD's1	289583ms	142846ms	6534560ms
With persistent RDD's2	301234ms	169234ms	7789089ms
Without using persistent RDD's 1	456765ms	234298ms	10989654ms
Without using persistent RDD's 1	509876ms	267890ms	12389654ms

I observed that persistent RDD's significantly decreased the processing time and improved the run time efficiency but persistent RDD's do consume a lot of memory.

Conclusion : Using K-means algorithm for geo location clustering is reliable but not perfect because k-means algorithm is not robust but it does give reliable results for certain k values but are not perfect because there are lot of disadvantages taking distance as a metric of measurement for geo location clustering .It would have performed well if the border separation is given. Learnt a lot about k-means and spark and using pyspark in the process of finishing the project .

Future Scope:

This project can be extended to some very interesting and useful analysis. With functionality of stream processing, Spark can be helpful to analyze stream data like Twitter and Foursquare. In stream data analysis, location clustering is useful way to compute the geo-location hot spot. More importantly, we can use these data to monitor what is happening around the world, to give much better event report (or prediction), safety reminder, advertisement, and so on.

Bibliography:

https://en.wikipedia.org/wiki/K-means_clustering

https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm

https://towardsdatascience.com/getting-started-with-pyspark-on-amazon-emr-c85154b6b921

https://en.wikipedia.org/wiki/Euclidean_distance

https://en.wikipedia.org/wiki/Great-circle_distance