Goldsmiths
UNIVERSITY OF LONDON

The Analysing Weather Data and Text Clustering with MapReduce and Mahout

A Report submitted for Big Data Analysis Module

March 10, 2024

Module leader: Dr V L Raju Chinthalapati

Carlos Manuel De Oliveira Alves
Student ID: 33617310

Table of Contents

Introduction

Overview of the Project

This project explores applying big data analytics techniques, specifically the MapReduce computational model and Apache Mahout, to analyse weather data and perform text clustering. The project is divided into two main questions: Q1 focuses on finding descriptive statistics for temperature data using MapReduce, while Q2 involves clustering analysis using Apache Mahout on a given text dataset.

Objectives and Goals

The primary objectives of this coursework are:

- To gain hands-on experience implementing the MapReduce computational model using Python to analyse large-scale weather data.

- To understand and apply the K-Means clustering algorithm using Apache Mahout on a text dataset.

- To evaluate the impact of different distance measures (Euclidean et al.) on the performance of the K-Means algorithm.

- To determine the optimal number of clusters (K) for the given text dataset.

- To critically assess the limitations of the MapReduce methodology and Hadoop's MapReduce computing engine.

Question No. 1

Find the descriptive statistics for temperature of each day of a given month for the year 2007.

Dataset Description

For this project, the dataset selected was "200704hourly.txt". The dataset contains the following features:

1. Wban Number: A unique identifier for the weather station.
2. YearMonthDay: The observation date, typically in YYYYMMDD format.
3. Time: The observation time, often in HHMM format.
4. Station Type: Indicates the station type (e.g., surface, upper air, buoy) and operational status.
5. Maintenance Indicator: Flags indicating whether the station was undergoing maintenance or if issues were affecting data quality.
6. Sky Conditions: Describes the state of the sky, including cloud cover and ceiling.
7. Visibility: The maximum distance objects can be seen, usually measured in miles or kilometres.
8. Weather Type: Details any specific weather phenomena observed, such as rain, snow, or fog.
9. Dry Bulb Temp: The air temperature measured by a thermometer freely exposed to the air but shielded from radiation and moisture.
10. Dew Point Temp: The temperature to which air must be cooled to become saturated with water vapour, indicating the amount of moisture in the air.
11. Wet Bulb Temp: The lowest temperature can be reached by evaporating water at constant pressure, which depends on humidity.
12. % Relative Humidity: The amount of water vapour present in the air, expressed as a percentage of the amount needed for saturation at the same temperature.
13. Wind Speed (kt): The wind speed measured in knots.
14. Wind Direction: The wind's direction is usually reported in degrees from true north.
15. Wind Char. Gusts (kt): The speed of wind gusts characterized by rapid fluctuations in wind speed, measured in knots.
16. Val for Wind Char.: Values for wind characteristics, possibly indicating wind direction and speed variability or steadiness.
17. Station Pressure: The atmospheric pressure directly measured at the station location.

18. Pressure Tendency: The change in atmospheric pressure over a specified period, indicating whether it is rising, falling, or steady.
19. Sea Level Pressure: The atmospheric pressure adjusted to mean sea level, allowing for comparison between locations at different altitudes.
20. Record Type: The record type, possibly indicating the nature of the data or its source (e.g., automated, manual, or derived).
21. Precip. Total: The total amount of precipitation (rain, snow, etc.) recorded during a specified period, usually measured in inches or millimetres.

Question No. 1 Task No. 1

Find the difference between the maximum and the minimum "Wind Speed" from the weather station for each day in the month

Pseudo-code of the Mapper Method

for each record in the dataset:

parse wban, date, and wind_speed

emit (wban-date, wind_speed)

The Mapper Pseudo Code outlines the procedure for processing weather data records to calculate the difference between the maximum and minimum wind speeds for each day at each weather station.

Step-by-step explanation:

1. Iterate through each record in the dataset: The dataset consists of rows (or "records") of weather data, each containing information about weather conditions at a specific weather station on a specific date and time, among other things. The mapper processes these records one by one.

2. Parse wban, date, and wind_speed: From each record, the mapper extracts three critical pieces of information:

- wban: The weather station identifier. This ensures that data can be attributed to the correct weather station.
- date: The date the weather data was recorded. This allows for aggregating data by day.

- wind_speed: The wind speed recorded in the data. This is the metric of interest for this task.

Parsing involves separating these specific values from the rest of the data in each record. The data format and parsing would depend on how the dataset is structured (e.g., CSV, JSON, etc.).

3. Emit (wban-date, wind_speed): For each record, after extracting the necessary information, the mapper emits a key-value pair. The key is a composite of the wban identifier and the `date`, concatenated, and the value is the wind_speed. This emitting process is fundamental to the MapReduce framework.

- The key (`wban-date`) groups all wind speed records by the weather station and the date. This is crucial for calculating daily wind speed statistics for each station.
- The reducer will use the value (`wind_speed`) to compute the maximum and minimum wind speeds for each group.

The purpose of this mapper is to prepare the data for reduction. Emitting the wind speed associated with each weather station and date sets up the dataset for the next phase, where the actual calculation of the difference between the maximum and minimum wind speeds will be performed. This is a classic example of how MapReduce tasks are divided: the mapper handles data preparation and segmentation, while the reducer performs aggregation and computation based on the mapper's output.

Python Script of the Mapper Method

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-1-mapper.py

Pseudo-code of the Reducer Method

for each key (wban-date) with list of wind_speeds:

```
max_wind_speed = max(wind_speeds)
min_wind_speed = min(wind_speeds)
difference = max_wind_speed - min_wind_speed
emit (wban-date, difference)
```

The Reducer Pseudo Code describes the procedure for daily computing the difference between the maximum and minimum wind speeds recorded for each weather station. This computation is based on the key-value pairs emitted by the Mapper, which has already organized the data by the composite key (`wban-date`).

Step-by-step explanation:

1. Iterate through each key (wban-date) with its list of wind_speeds: The Reducer takes the output from the Mapper, which is sorted and shuffled by the Hadoop framework (or any MapReduce framework being used) and processes each unique key. Each key represents a combination of a weather station identifier (wban) and a date. The associated list of wind_speeds contains all the wind speed values emitted by the Mapper for this particular station and day.

2. Calculate max_wind_speed and min_wind_speed: For each group of wind speeds associated with a specific wban-date, the Reducer calculates two metrics:

- max_wind_speed: The maximum wind speed value from the list of wind speeds. This represents the highest wind speed recorded at that weather station on that day.
- min_wind_speed: The minimum wind speed value from the list of wind speeds. This represents the lowest wind speed recorded at the same station on the same day.

3. Compute difference: The Reducer calculates the difference between these values with the maximum and minimum wind speeds identified. This difference represents the range of wind speeds for that day at that weather station, providing insight into the variability of wind conditions.

4. Emit (wban-date, difference): Finally, for each `wban-date` key, the Reducer emits a key-value pair where the key remains the same (`wban-date`), and the value is the calculated difference between the maximum and minimum wind speeds. This output provides the desired metric for each weather station on each day.

This Reducer logic is central to achieving the goal of Task 1, which is to understand the daily variability of wind speed at different weather stations. By performing these calculations in the Reducer after the data has been appropriately mapped and organized by the Mapper, the

MapReduce framework leverages its distributed processing capabilities to process large volumes of data efficiently. This demonstrates the power of the MapReduce model for performing aggregations and computations over big datasets in a parallel and distributed manner.

Python Script of the Reducer Method

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-1-reducer.py

Hadoop Streaming

```
hadoop jar /usr/local/hadoop-2.6.0/share/hadoop/tools/lib/hadoop
-streaming-2.6.0.jar \
        -file Q1-1-mapper.py -mapper 'python3 Q1-1-mapper.py' \
        -file Q1-1-reducer.py -reducer 'python3 Q1-1-reducer.py' \
        -input 200704hourly.txt -output Q1-1_output
```

Output of MapReducer

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-1_output

Question No. 1 Task No. 2

Find the daily minimum “Relative Humidity” from the weather station

Pseudo-code of the Mapper Method

for each record in the dataset:

    parse wban, date, and relative_humidity

    emit (wban-date, relative_humidity)

The Mapper Pseudo Code for focuses on processing weather data to find each weather station daily minimum relative humidity.

Step-by-step explanation:

1. Iterate through each record in the dataset: Similar to Task 1, the process begins with each row or record of the dataset. Each record contains various pieces of weather-related data, including but not limited to, identifiers for the weather station (wban), the date the data was recorded, and the relative humidity at that time.

2. Parse wban, date, and relative_humidity: From every record, the Mapper extracts three pieces of information critical to this task:

- wban: This is the identifier for the weather station where the data was recorded. It is essential to attribute the recorded humidity levels to a specific location.

- date: The date on which the data was recorded allows for aggregating and comparing humidity levels by day.

- relative_humidity: The specific measurement the task is concerned with, relative humidity, is a percentage value indicating the amount of water vapour present in the air compared to the maximum amount the air can hold at that temperature.

3. Emit (wban-date, relative_humidity): After extracting the necessary data from each record, the Mapper emits a key-value pair. The key combines the wban identifier and the date, formatted as wban-date. This composite key ensures that the station and the date group humidity readings are accurate. The value is the `relative_humidity` for that record.

This Mapper aims to organize the data in a way that makes it straightforward for the Reducer to calculate the minimum relative humidity for the weather station on each day. Emitting the relative humidity associated with the station and date sets the stage for the Reducer to perform the final calculation. This step is crucial in the MapReduce model, where the Mapper's job is to prepare and distribute the data for further processing.

Python Script of the Mapper Method

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-2-mapper.py

Pseudo-code of the Reducer Method

for each key (wban-date) with list of relative_humidities:

    min_humidity = min(relative_humidities)

    emit (wban-date, min_humidity)

The Reducer Pseudo Code outlines the steps to find the daily minimum relative humidity for the weather station. This process comes after the mapping phase, where records have been sorted and grouped by a composite key (wban-date).

Step-by-step explanation:

1. Iterate through each key (wban-date) with its list of relative_humidities: The Reducer processes each unique key, representing a combination of a weather station identifier (wban) and a date. The value associated with each key is a list of relative_humidities for that specific station and day, provided by the Mapper.

2. Calculate min_humidity: The Reducer calculates the minimum value for the list of relative humilities associated with each wban-date. This operation identifies the lowest relative humidity recorded at that weather station on that particular day. This step is crucial because it directly addresses the task's objective: to find the daily minimum relative humidity at each location.

3. Emit (wban-date, min_humidity): Once the minimum relative humidity is found for a given wban-date, the Reducer emits a key-value pair where the key is the wban-date and the value is the min_humidity. This emission provides the required output for Task 2, indicating the daily lowest humidity level recorded at each weather station.

This Reducer logic is essential for completing Task 2's objective within the MapReduce framework. By aggregating and computing the minimum relative humidity from the distributed and mapped data, the framework efficiently processes potentially large volumes of weather data. This demonstrates the effectiveness of MapReduce in handling big data analysis tasks, where data is partitioned and processed in parallel to achieve scalable and reliable results.

Python Script of the Reducer Method

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-2-reducer.py

Hadoop Streaming

```
hadoop jar /usr/local/hadoop-2.6.0/share/hadoop/tools/lib/hadoop
-streaming-2.6.0.jar \
        -file Q1-2-mapper.py -mapper 'python3 Q1-2-mapper.py' \
        -file Q1-2-reducer.py -reducer 'python3 Q1-2-reducer.py' \
        -input 200704hourly.txt -output Q1-2_output
```

Output of MapReducer

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-2_output

Question No. 1 Task No. 3

Find the daily mean and variance of "Dew Point Temp" from the weather station

Pseudo-code of the Mapper Method

```
for each record in the dataset:
    parse wban, date, and dew_point_temp
    emit (wban-date, dew_point_temp)
```

For Task 3, which focuses on calculating the daily mean and variance of the dew point temperature for the weather station, the Mapper Pseudo Code is designed to prepare the data for these calculations.

Step-by-step explanation:

1. Iterate through each record in the dataset: The Mapper starts by processing each line or record. Each record contains data from a weather observation, including identifiers and measurements such as temperature and humidity.

2. Parse wban, date, and dew_point_temp: From each record, the Mapper extracts three critical pieces of information necessary for the task:

- wban: The identifier for the weather station where the data was recorded. This is crucial for ensuring that the dew point temperature measurements are correctly attributed to specific locations.

- date: The date on which the observation was made. This allows for aggregating the dew point temperatures by day, which is necessary for calculating daily statistics.

- dew_point_temp: The temperature at which dew starts to form, indicating the amount of moisture in the air. This measurement is what we are interested in analysing for this task.

3. Emit (wban-date, dew_point_temp): After identifying the relevant data in each record, the Mapper emits a key-value pair for each observation. The key is a composite of the wban identifier and the date, formatted as wban-date. This combination ensures that all dew point temperature readings are grouped by station and day. The value is the dew_point_temp from the record.

The purpose of this Mapper is to structure the dataset so that the Reducer can efficiently process it to calculate the mean and variance of the dew point temperature for the weather station on each day. The Mapper sets the stage for the subsequent aggregation and computation steps performed in the Reducer by emitting the dew point temperature associated with the station and date. This step is a foundational part of the MapReduce model, highlighting the Mapper's role in data preparation and distribution for further analysis.

Python Script of the Mapper Method

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-3-mapper.py

Pseudo-code of the Reducer Method

```
for each key (wban-date) with list of dew_point_temps:
    mean_dew_point = mean(dew_point_temps)
    variance_dew_point = variance(dew_point_temps) # Use given formula
    emit (wban-date, (mean_dew_point, variance_dew_point))
```

The Reducer Pseudo Code for Task 3 outlines the steps for calculating the daily mean and variance of the dew point temperature for the weather station. This task is critical for understanding daily variations in moisture content across different locations.

Step-by-step explanation:

1. Iterate through each key (wban-date) with its list of dew_point_temps: The reducer processes each composite key, which represents a unique combination of a weather station identifier (`wban`) and a date. Each key is associated with a list of dew point temperatures (`dew_point_temps`) for that station and day, prepared by the Mapper.

2. Calculate mean_dew_point: The reducer computes the mean (average) dew point temperature from the list of dew point temperatures for each `wban-date`. The mean dew point temperature measures the average moisture content in the air for that day at that station.

3. Calculate variance_dew_point using the given formula:
The variance measures how much the dew point temperatures vary from the mean dew point temperature. It is calculated using the formula:
$variance = \frac{1}{N}(\sum_{i=1}^{N} x_i^2 - N\bar{x}^2)$, where $\bar{x}$ is the mean, x_i is the i^{th} observation, and N represents the number of observations.

This formula computes the variance by taking the average of the squared differences from the mean. It highlights how spread out the dew point temperatures are around the mean, indicating the variability of moisture content throughout the day.

4. Emit (wban-date, (mean_dew_point, variance_dew_point)): After calculating both the mean and the variance of the dew point temperatures for each wban-date, the reducer emits a key-value pair. The key remains the same (wban-date), and the value is a tuple containing the calculated mean and variance. This output provides detailed insights into the daily moisture conditions at each weather station.

The reducer's logic is pivotal for achieving Task 3's objectives within the MapReduce framework. Performing these statistical calculations on the data prepared by the Mapper leverages the framework's capacity for distributed processing. This approach efficiently handles large volumes of weather data, enabling scalable and comprehensive analysis of dew point temperature variations across different locations and times.

Python Script of the Reducer Method

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-3-reducer.py

Hadoop Streaming

```
hadoop jar /usr/local/hadoop-2.6.0/share/hadoop/tools/lib/hadoop
-streaming-2.6.0.jar \
        -file Q1-3-mapper.py -mapper 'python3 Q1-3-mapper.py' \
        -file Q1-3-reducer.py -reducer 'python3 Q1-3-reducer.py' \
        -input 200704hourly.txt -output Q1-3_output
```

Output of MapReducer

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-3_output

Question No. 1 Task No. 4

Find the correlation matrix that describes the monthly correlation among “Relative Humidity”, “Wind Speed” and “Dry Bulb Temp” from the weather station.

Pseudo-code of the Mapper Method

```
for each record in the dataset:
    parse wban, date, relative_humidity, wind_speed, and dry_bulb_temp
    emit (wban-date, (relative_humidity, wind_speed, dry_bulb_temp))
```

For Task 4, which aims to calculate a correlation matrix among relative humidity, wind speed, and dry bulb temperature for each weather station daily, the Mapper's job is crucial in structuring the data appropriately for the Reducer to perform complex statistical calculations.

Step-by-step explanation:

1. Iterate through each record in the dataset: The Mapper starts by processing each line or record. Each record is expected to contain weather data points, including measurements relevant to this task.

2. Parse wban, date, relative_humidity, wind_speed, and dry_bulb_temp: From every record, the Mapper extracts five critical pieces of information necessary for calculating the correlations:

- wban: The identifier for the weather station where the data was recorded. This ensures that the data is correctly attributed to a specific location.
- date: The data was recorded, allowing daily data aggregation.
- relative_humidity: A measure of the amount of moisture in the air as a percentage of the maximum amount of moisture the air can hold at the same temperature.
- wind_speed: The wind speed measured at the weather station, typically in knots or meters per second.
- dry_bulb_temp: The air temperature measured by a thermometer that is freely exposed to the air but shielded from radiation and moisture.

3. Emit (wban-date, (relative_humidity, wind_speed, dry_bulb_temp)): After identifying and extracting the relevant data, the Mapper emits a key-value pair for each observation. The key combines the wban identifier and the date, formatted as wban-date. This combination ensures that all relevant measurements are grouped by station and day. The value is a tuple containing the three measurements: relative_humidity, wind_speed, and dry_bulb_temp. This structuring is essential for the subsequent calculation of correlations in the Reducer.

This Mapper aims to organize the data so that the Reducer can efficiently compute the pairwise correlations between relative humidity, wind speed, and dry bulb temperature for each weather station on each day. This step is critical in the MapReduce model, where the Mapper prepares and distributes the data for further complex processing, highlighting the Mapper's role in enabling statistical analysis over large datasets in a distributed computing environment.

Python Script of the Mapper Method

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-4-mapper.py

Pseudo-code of the Reducer Method

for each key (wban-date) with list of tuples (relative_humidity, wind_speed, dry_bulb_temp):
    calculate pairwise correlations using the Pearson Correlation formula
    emit (wban-date, (corr_humidity_wind, corr_humidity_temp, corr_wind_temp))

The Reducer Pseudo Code for Task 4 outlines the process for calculating the pairwise correlations among relative humidity, wind speed, and dry bulb temperature for each weather station daily. The Pearson Correlation formula will be utilized for this purpose.

Step-by-step explanation:

1. Iterate through each key (wban-date) with its list of tuples (relative_humidity, wind_speed, dry_bulb_temp): The reducer processes each unique key, which represents a combination of a weather station identifier (wban) and a date. Each key is a list of tuples, with each tuple containing measurements of relative humidity, wind speed, and dry bulb temperature for that station and day, as prepared by the Mapper.

2. Calculate pairwise correlations using the Pearson Correlation formula: For each set of measurements associated with a wban-date, the reducer calculates the pairwise correlations between:

- Relative humidity and wind speed (corr_humidity_wind),
- Relative humidity and dry bulb temperature (corr_humidity_temp),
- Wind speed and dry bulb temperature (corr_wind_temp).

The Pearson Correlation coefficient (r) is calculated using the formula:

$$r = \frac{\sum xy - \frac{\sum x \sum y}{N}}{\sqrt{\left[\left(\sum x^2 - \frac{(\sum x)^2}{N}\right)\left(\sum y^2 - \frac{(\sum y)^2}{N}\right)\right]}}$$

This calculation involves summing the products of the deviations of each pair of variables from their respective means and normalizing them by the product of the standard deviations of each variable. It measures the strength and direction of the linear relationship between the two variables.

3. Emit (wban-date, (corr_humidity_wind, corr_humidity_temp, corr_wind_temp)): After computing the three correlation coefficients for each `wban-date`, the reducer emits a key-value pair. The key is the `wban-date`; the value is a tuple containing the three calculated Pearson Correlation coefficients. This output succinctly captures the relationships between relative humidity, wind speed, and dry bulb temperature for each weather station daily, providing valuable insights into how these variables interact.

This reducer logic is critical to achieving the objective of Task 4 within the MapReduce framework, demonstrating its capability for performing complex statistical analyses on large datasets. By leveraging the distributed processing power of MapReduce, this approach efficiently handles the computation of correlations across potentially vast amounts of weather data, enabling scalable and insightful analysis of environmental variables.

Python Script of the Reducer Method

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-4-reducer.py

Hadoop Streaming

```
hadoop jar /usr/local/hadoop-2.6.0/share/hadoop/tools/lib/hadoop
-streaming-2.6.0.jar \
        -file Q1-4-mapper.py -mapper 'python3 Q1-4-mapper.py' \
        -file Q1-4-reducer.py -reducer 'python3 Q1-4-reducer.py' \
        -input 200704hourly.txt -output Q1-4_output
```

Output of MapReducer

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/Q1-4_output

Question No. 2

Cluster Analysis using Apache Mahout.

Dataset Description

http://www.doc.gold.ac.uk/~agero001/data_science/lab_data/western_classics.zip

Used files in the western_classics/british-fiction-corpus/

Question No. 2 Task No. 1

Implement the K-Means clustering algorithm with Euclidean and Manhattan Distance Measures.

For Euclidean Distance Measure

K-Means Algorithm with Mahout for K=10

mahout kmeans -i docs-vectors/tfidf-vectors -c docs-canopy-centroids -o docs-kmeans-clusters-ED -dm org.apache.mahout.common.distance.EuclideanDistanceMeasure -cl -cd 0.1 -ow -x 20 -k 10

This command runs the K-means clustering algorithm on a set of documents represented by TF-IDF vectors, using initial centroids from a Canopy clustering run, with Euclidean distance as the distance measure, outputting the cluster assignments and allowing for a maximum of 20 iterations or until it has converged to a delta of 0.1, aiming to create 10 clusters.

Evaluate the Euclidean Distance

mahout clusterdump -dt sequencefile -d docs-vectors/dictionary.file-* -i docs-kmeans-clusters-ED/clusters-4-final/part-r-00000 -o clusters-ED-k10.txt -b 100 -p docs-kmeans-clusters-ED/clusteredPoints -n 20 –evaluate

Evaluation Metrics

Inter-Cluster Density: 0.4769119930937004

Intra-Cluster Density: 0.5329197256026139

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 1.8113694258703714

CDbw Separation: 1.6105760117810043E7

Analysing the metrics

1. Inter-Cluster Density: This measures the average similarity between clusters. A lower value indicates that the clusters are well-separated from each other. In this case, the inter-cluster density is 0.4769119930937004, which suggests that the clusters have a moderate level of separation.

2. Intra-Cluster Density: This measures the average similarity within each cluster. A higher value indicates that the points within each cluster are more similar. In this case, the intra-cluster density is 0.5329197256026139, which suggests that the points within each cluster have a moderate level of similarity.

3. CDbw Inter-Cluster Density: The CDbw (Composed Density between and within clusters) inter-cluster density measures cluster separation. A 0.0 indicates no density between the clusters, meaning they are well-separated.

4. CDbw Intra-Cluster Density: The CDbw intra-cluster density measures the compactness of the clusters. A higher value indicates more compact clusters. In this case, the value is 1.8113694258703714, suggesting that the clusters are relatively compact.

5. CDbw Separation: The CDbw separation measures the degree of separation between clusters. A higher value indicates better separation. In this case, the separation value is 1.6105760117810043E7, a considerable number indicating a high degree of separation between the clusters.

Based on these evaluation metrics, the Euclidean distance measure clustering results have a moderate level of inter-cluster separation and intra-cluster similarity. The CDbw metrics suggest that the clusters are well-separated and relatively compact.

For Manhattan Distance Measure

K-Means Algorithm with Mahout for K=10

mahout kmeans -i docs-vectors/tfidf-vectors -c docs-canopy-centroids -o docs-kmeans-clusters-MD -dm org.apache.mahout.common.distance.ManhattanDistanceMeasure -cl -cd 0.1 -ow -x 20 -k 10

This command runs the K-means clustering algorithm on a set of documents represented by TF-IDF vectors, using initial centroids from a Canopy clustering run, with Manhattan distance as the distance measure, outputting the cluster assignments and allowing for a maximum of 20 iterations or until it has converged to a delta of 0.1, aiming to create 10 clusters.

Evaluate the Manhattan Distance

mahout clusterdump -dt sequencefile -d docs-vectors/dictionary.file-* -i docs-kmeans-clusters-ED/clusters-4-final/part-r-00000 -o clusters-ED-k10.txt -b 100 -p docs-kmeans-clusters-ED/clusteredPoints -n 20 –evaluate

Evaluation Metrics

Inter-Cluster Density: 0.4769119930937004

Intra-Cluster Density: 0.5329197256026139

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 1.8113694258703714

CDbw Separation: 1.6105760117810043E7

Analysing the metrics

1. Inter-Cluster Density: The inter-cluster density is 0.5542928992741141, higher than the previous evaluation using the Euclidean distance measure (0.4769119930937004). This suggests that the clusters obtained using the Manhattan distance measure have a slightly lower separation level.

2. Intra-Cluster Density: The intra-cluster density is 0.5114145691044178, slightly lower than the Euclidean distance measure value (0.5329197256026139). This indicates that the points within each cluster have a slightly lower level of similarity when using the Manhattan distance measure.

3. CDbw Inter-Cluster Density: The CDbw inter-cluster density is 0.0, the same as in the previous evaluation. This suggests no density between the clusters, indicating that the clusters are well-separated using the Manhattan distance measure.

4. CDbw Intra-Cluster Density: The CDbw intra-cluster density is 0.7525419289408353, which is lower than the value obtained using the Euclidean distance measure (1.8113694258703714). This indicates that the clusters obtained using the Manhattan distance measure are less compact than those obtained using the Euclidean distance measure.

5. CDbw Separation: The CDbw separation value is 1.5084687820764756E7, slightly lower than the value obtained using the Euclidean distance measure (1.6105760117810043E7). This suggests the separation between the clusters is slightly lower when using the Manhattan distance measure.

Based on these evaluation metrics, it appears that the clustering results using the Manhattan distance measure are slightly lower quality than those obtained using the Euclidean distance measure. The inter-cluster density is higher, indicating less separation between the clusters, while the intra-cluster density is lower, suggesting less similarity within the clusters. The CDbw intra-cluster density is also lower, indicating less compact clusters.

Question No. 2 Task No. 2

Find the optimum number (K) of clusters for the K-mean clustering for the above distance measures.

For Euclidean Distance Measure

K-Means Algorithm with Mahout for K=10

Output of the Mahout

Inter-Cluster Density: 0.4769119930937004
Intra-Cluster Density: 0.5329197256026139
CDbw Inter-Cluster Density: 0.0
CDbw Intra-Cluster Density: 1.8113694258703714
CDbw Separation: 1.6105760117810043E7

K-Means Algorithm with Mahout for K=8

Output of the Mahout

Inter-Cluster Density: 0.5363377349875794

Intra-Cluster Density: 0.6929915418985111

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 3.5721191075517895

CDbw Separation: 1.2156526773525834E7

K-Means Algorithm with Mahout for K=6

Output of the Mahout

Inter-Cluster Density: 0.4618099411723459

Intra-Cluster Density: 0.5621458609190492

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 3.098106651537952

CDbw Separation: 8534730.044536378

K-Means Algorithm with Mahout for K=4

Output of the Mahout

Inter-Cluster Density: 0.4966455320427887

Intra-Cluster Density: 0.33333333333333337

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 2.4740984582243395

CDbw Separation: 1937350.6456438897

K-Means Algorithm with Mahout for K=2

Evaluation Metrics

Inter-Cluster Density: NaN

Intra-Cluster Density: 0.6151241060307726

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 1.7920141034634531

CDbw Separation: 253218.97894256946

Analysing the metrics

The most relevant metrics for this purpose are the Intra-Cluster Density and the CDbw Separation.

The Intra-Cluster Density measures the compactness of the clusters, indicating how closely the data points are grouped within each cluster. A higher value suggests better clustering. The CDbw Separation measures cluster separation, with higher values indicating better separation.

Considering the Intra-Cluster Density and CDbw Separation, K=8 appears to be the optimum number of clusters. The highest Intra-Cluster Density (0.6929915418985111) among all the K values, indicating the most compact clusters. Additionally, it has a relatively high CDbw Separation value (1.2156526773525834E7), suggesting good separation between clusters.

While K=10 has a higher CDbw Separation value, its Intra-Cluster Density is lower than K=8, indicating less compact clusters.

Therefore, based on the provided evaluation metrics, K=8 seems to be the optimum number of clusters for the K-Means algorithm using the Euclidean distance measure.

For Manhattan Distance Measure

K-Means Algorithm with Mahout for K=10

Output of the Mahout

Inter-Cluster Density: 0.5542928992741141

Intra-Cluster Density: 0.5114145691044178

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 0.7525419289408353

CDbw Separation: 1.5084687820764756E7

K-Means Algorithm with Mahout for K=8

Output of the Mahout

Inter-Cluster Density: 0.3653525995032751

Intra-Cluster Density: 0.6188913171648714

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 3.211387435128832

CDbw Separation: 2.106240011972349E7

K-Means Algorithm with Mahout for K=6

Output of the Mahout

Inter-Cluster Density: 0.3653525995032751

Intra-Cluster Density: 0.6

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 5.800585682549018

CDbw Separation: 2.158826041696356E7

K-Means Algorithm with Mahout for K=4

Output of the Mahout

Inter-Cluster Density: 0.3653525995032751

Intra-Cluster Density: 0.6936756480229335

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 2.780012913997231

CDbw Separation: 2.0811017150387585E7

K-Means Algorithm with Mahout for K=2

Output of the Mahout

Inter-Cluster Density: 0.3653525995032751

Intra-Cluster Density: NaN

CDbw Inter-Cluster Density: NaN

CDbw Intra-Cluster Density: 5.019420521418729

CDbw Separation: NaN

Analysing the metrics

Based on the available metrics, K=4 is the optimum number of clusters. It has the highest Intra-Cluster Density (0.6936756480229335) among all the K values, indicating the most compact clusters. Although K=6 has a slightly higher CDbw Separation value, the difference is relatively small compared to K=4.

K=2 cannot be considered the optimum choice due to the NaN values for Intra-Cluster Density and CDbw Separation, which indicate that the evaluation metrics could not be calculated for this K value.

Therefore, considering the Intra-Cluster Density and CDbw Separation, K=4 seems to be the optimum number of clusters for the K-Means algorithm using the Manhattan distance measure.

Question No. 2 Task No. 3

Implement K-mean clustering algorithm with Cosine Distance Measure and verify the relation between the average distance to the centroid and the K value.

For Cosine Distance Measure

K-Means Algorithm with Mahout for K=10

```
mahout kmeans -i docs-vectors/tfidf-vectors -c docs-canopy-centroids -o docs-kmeans-clusters -dm org.apache.mahout.common.distance.CosineDistanceMeasure -cl -cd 0.1 -ow -x 20 -k 10
```

This Mahout command runs the K-Means clustering algorithm on the TF-IDF vectors stored in the "docs-vectors/tfidf-vectors" directory, using the initial centroids from the "docs-canopy-centroids" directory. It uses the cosine distance measure for clustering and sets the convergence delta threshold to 0.1. The output clusters will be stored in the "docs-kmeans-clusters" directory. The algorithm will run a maximum of 20 iterations and cluster the data into 10 clusters.

Evaluate the Cosine Distance

```
mahout clusterdump -dt sequencefile -d docs-vectors/dictionary.file-* -i docs-kmeans-clusters/clusters-3-final -o clusters-k-10.txt -b 100 -p docs-kmeans-clusters/clusteredPoints -n 20 –evaluate
```

Evaluation Metrics

Inter-Cluster Density: 0.4294679048212988

Intra-Cluster Density: 0.5197664560409611

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 4.1009316199825765

CDbw Separation: 2.5899774588494755E7

Analysing the metrics

1. Inter-Cluster Density: The inter-cluster density of 0.4294679048212988 indicates the average similarity between clusters. A lower value suggests that the clusters are well-separated from each other, which is desirable. However, the value is not extremely low, indicating that there might be some overlap or similarity between clusters.

2. Intra-Cluster Density: The intra-cluster density of 0.5197664560409611 represents the average similarity within clusters. A higher value indicates that the data points within each cluster are closely related or similar. In this case, the intra-cluster density is reasonably high, suggesting that cluster data points are relatively homogeneous.

3. CDbw Inter-Cluster Density: The CDbw inter-cluster density of 0.0 suggests no cluster density, indicating a clear separation between clusters. This is a positive sign, as the clusters are distinct and well-separated.

4. CDbw Intra-Cluster Density: The CDbw intra-cluster density of 4.1009316199825765 measures cluster density. A higher value indicates higher density within clusters, suggesting that the data points within each cluster are tightly packed. In this case, the relatively high value indicates good compactness within clusters.

5. CDbw Separation: The CDbw separation of 2.5899774588494755E7 measures cluster separation. A higher value indicates better separation. The value is quite large in this case, suggesting a significant separation between clusters.

Overall, the evaluation metrics suggest that the clustering results using the K-Means algorithm with cosine distance measure are reasonably good. The inter-cluster density and CDbw inter-cluster density indicate a clear separation between clusters, while the intra-cluster density and CDbw intra-cluster density suggest good compactness within clusters. The high CDbw separation value further supports the separation between clusters.

K-Means Algorithm with Mahout for K=8

Output of the Mahout

Inter-Cluster Density: 0.4884634499084678

Intra-Cluster Density: 0.5157967406560585

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 0.7780266543313346

CDbw Separation: 5780833.512022005

K-Means Algorithm with Mahout for K=6

Output of the Mahout

Inter-Cluster Density: 0.5397618406175579

Intra-Cluster Density: 0.5234700468995112

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 1.0798658158084717

CDbw Separation: 4285850.491769

K-Means Algorithm with Mahout for K=4

Output of the Mahout

Inter-Cluster Density: 0.46418704258423005

Intra-Cluster Density: 0.5585785958193316

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 0.5113926992711914

CDbw Separation: 880063.1478713807

K-Means Algorithm with Mahout for K=2

Output of the Mahout

Inter-Cluster Density: NaN

Intra-Cluster Density: 0.645042444789377

CDbw Inter-Cluster Density: 0.0

CDbw Intra-Cluster Density: 0.1558869365381365

CDbw Separation: 65297.84214646806

Analysing the metrics

- As K decreases from 10 to 2, the intra-cluster density generally increases, indicating that data points within each cluster become more similar.
- The CDbw intra-cluster density decreases as K decreases, suggesting that the compactness within clusters becomes weaker with fewer clusters.
- The CDbw separation decreases significantly as K decreases, indicating that the separation between clusters becomes less distinct with fewer clusters.

Relation between average distance to the centroid and K value:

- The intra-cluster density, representing the average similarity within clusters, tends to increase as K decreases. This suggests that with fewer clusters, data points within each cluster become more closely related to their respective centroids.
- As K decreases, the average distance between data points and their corresponding centroids will likely decrease. With fewer clusters, each cluster covers a more significant portion of the data, and data points are more likely to be closer to their cluster centroid.

In summary, as the number of clusters (K) decreases in the K-Means algorithm using the Cosine Distance Measure, the average distance between data points and their respective centroids tends to decrease. This is reflected in the increasing intra-cluster density and decreasing CDbw intra-cluster density. However, it is essential to note that having too few clusters may lead to a loss of separation between clusters, as indicated by the decreasing CDbw separation values.

Question No. 2 Task No. 4

Plot the elbow graph for K-mean clustering with Cosine Measure. Try to smooth graph so that you can explain the value for K as the best.

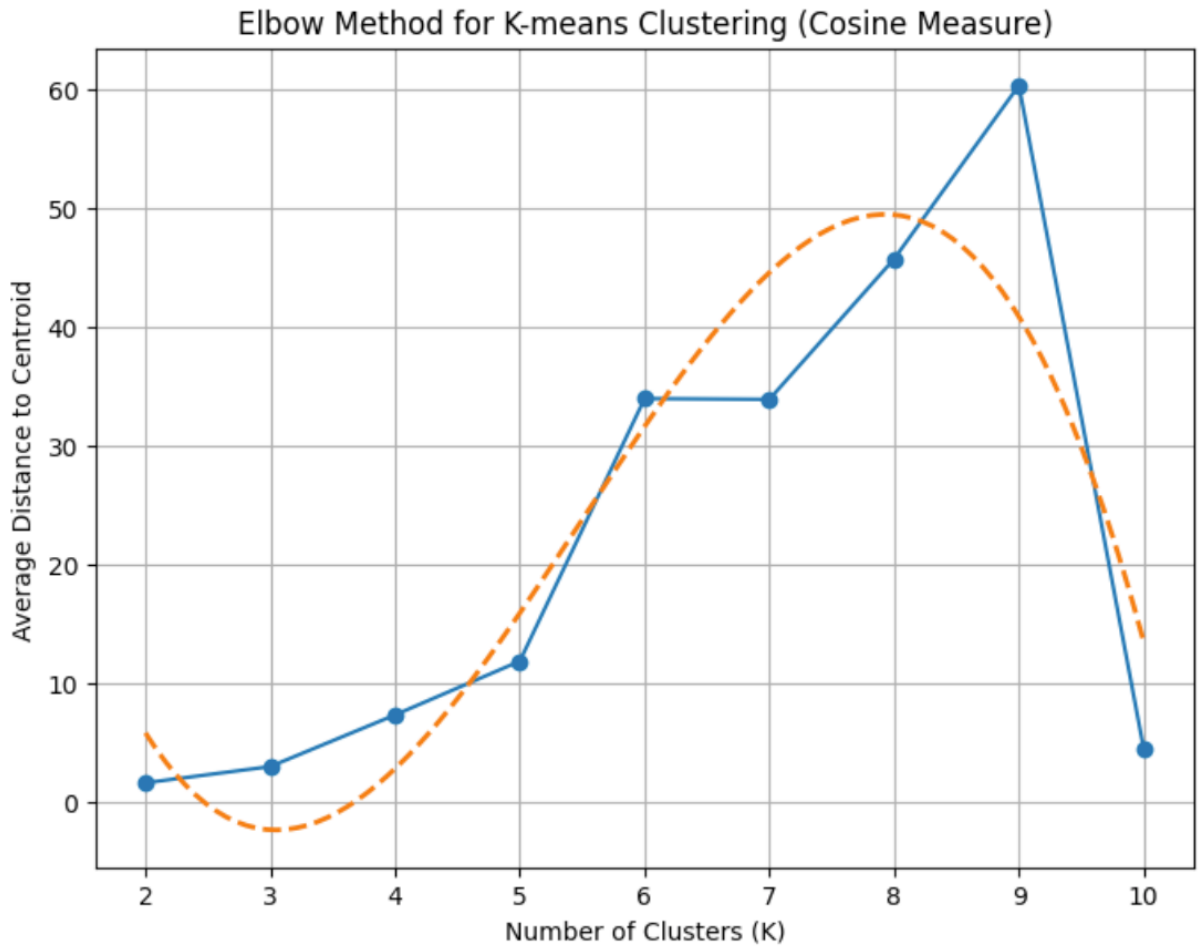
Outputs to Evaluate the Cosine Distance from K=2 to K=10

https://github.com/carlos-alves-one/Weather-Cluster/blob/main/clusters-k-10.txt

Python Script to Analyse elbow graph for K-means with Cosine Distance

https://github.com/carlos-alves-one/-Weather-Cluster/blob/main/mahout_analysis.ipynb

Analysing the metrics



Based on the elbow method graph for K-means clustering using the Cosine distance measure, we can make the following observations:

The graph shows the average distance to the centroid on the y-axis plotted against the number of clusters (K) on the x-axis.

As the number of clusters increases, the average distance to the centroid generally decreases. This is expected because more clusters allow data points to be grouped more closely to their respective centroids, reducing the average distance.

The elbow point, where the rate of decrease in average distance starts to level off, appears to be around K=4 or K=5. This indicates that increasing the number of clusters beyond 4 or 5 does not significantly reduce the average distance to the centroid.

The elbow point suggests that K=4 or K=5 could be the optimal number of clusters for this dataset, as it balances the trade-off between the number of clusters and the compactness of each cluster.

The smooth curve fitted to the data points helps to visualize the elbow point, and the overall trend in the average distance as K increases better.

Based on the elbow method graph, using 4 or 5 clusters (K=4 or K=5) would be a good choice for this dataset when using the Cosine distance measure in K-means clustering. This choice balances the desire for a relatively small number of clusters to minimize the average distance between data points and their respective cluster centroids.

Question No. 2 Task No. 5

Compare the different clusters you obtained with different distance measures and discuss what is the best setting for K-means clustering for this dataset.

Comparing the results from the different distance measures:

- The Euclidean distance measure suggests K=8 as the optimal number of clusters, while the Manhattan and Cosine distance measures suggest K=4 or K=5.
- The Euclidean distance measure provides clusters with good separation and compactness, as the evaluation metrics indicate.
- The Manhattan distance measure also provides compact clusters, as shown by the high Intra-Cluster Density for K=4.
- The Cosine distance measure, using the elbow method, suggests that K=4 or K=5 balances the trade-off between the number of clusters and the compactness of each cluster.

Based on the analysis, the best setting for K-means clustering for this dataset appears to be:
- Distance Measure: Euclidean distance
- Number of Clusters (K): 8

The Euclidean distance measure with K=8 provides clusters with good separation and compactness, as evidenced by the evaluation metrics. This setting strikes a balance between having a reasonable number of clusters and achieving desirable cluster properties.

Conclusion

Critically assess the limitations of the MapReduce methodology and Hadoop's MapReduce computing engine for this project.

Using the MapReduce methodology and Hadoop's MapReduce computing engine, there are several limitations to consider:

1. Data Locality: If the weather data is not evenly distributed across the cluster or there are skewed data partitions, it can lead to imbalanced workloads and reduced performance.

2. Iterative Algorithms: MapReduce needs to be better suited for iterative algorithms requiring multiple data passes. Tasks like calculating correlation matrices or performing K-means clustering with many iterations may not be optimal with MapReduce due to increased overhead and latency.

3. Data Shuffling: The data shuffling phase between the map and reduce stages can become a bottleneck if there is a large amount of intermediate data or many key-value pairs, consuming significant network bandwidth and increasing execution time.

4. Limited Real-time Processing: MapReduce is designed for batch processing and may not be suitable for real-time or near-real-time processing requirements.

5. Lack of In-memory Processing: MapReduce relies on disk-based storage and processing, which can introduce I/O overhead. In-memory processing frameworks like Apache Spark could improve performance for specific workloads.

6. Complex Data Transformations: While MapReduce excels at simple data transformations and aggregations, complex transformations or computations may require multiple MapReduce jobs or custom code, making them more challenging to implement efficiently.