BDS Assignment - MapReduce

Group-4

2024mt03579@wilp.bits-pilani.ac.in	Deviprasad Tummidi
2024mt03613@wilp.bits-pilani.ac.in	Sandeep Kumar Mishra
2024mt03611@wilp.bits-pilani.ac.in	Aditya Jambhalikar
2024mt03554@wilp.bits-pilani.ac.in	Srivatsa D

Table of contents

Problem Statement

Analysis of News Data: Verifying Source Credibility and Truthfulness

This project will analyze a dataset of thousands of fact-checked news headlines to analyse the following points

Analysis to be performed:

1. Veracity count e.g. Number of false news, true news and other categories
2. Top 3 sources of most false and true statements e.g. News, social media etc

3. Top 3 originators of false and true statements which could be a person or post
4. Top 3 Month-Year with most false news by count and percentage
5. Top 5 keywords found in false news headlines

Dataset & Source Information

The dataset has been obtained from Kggle

https://www.kaggle.com/datasets/rmisra/politifact-fact-check-dataset

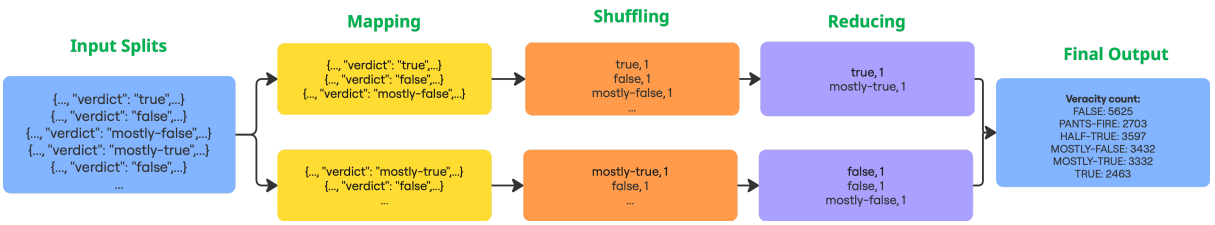
The data has been gathered from a website PolitiFact which fact checks the news. The news statements has been categorised into 6 categories: true, mostly true, half true, mostly false, false, and pants on fire

It has more than 21k news headlines fact checked

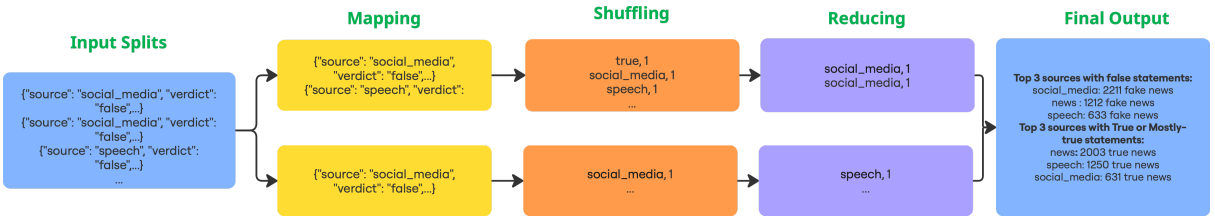
Source of dataset is rishabhmisra.github.io/publications

Map-Reduce Diagrams for each analysis task

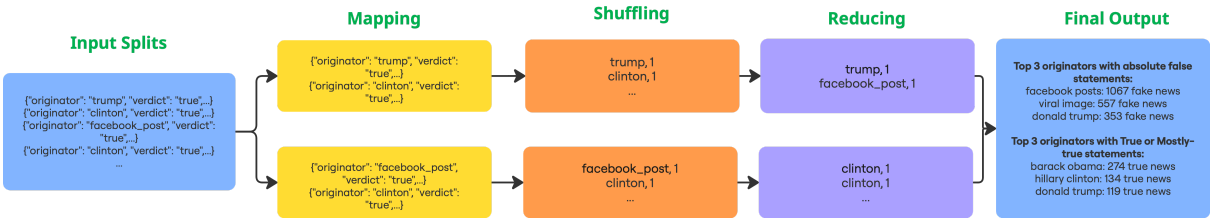
Veracity Count



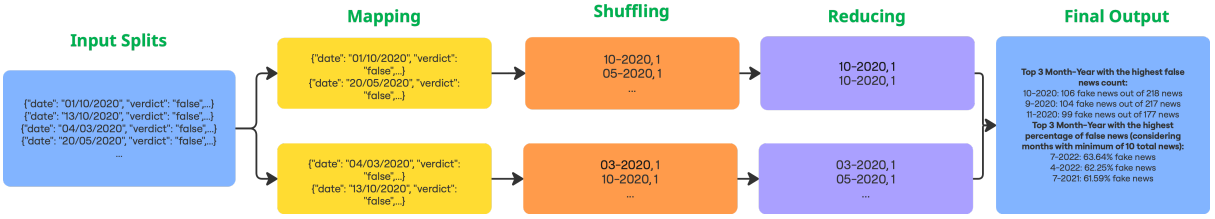
Top Sources with false/true statements



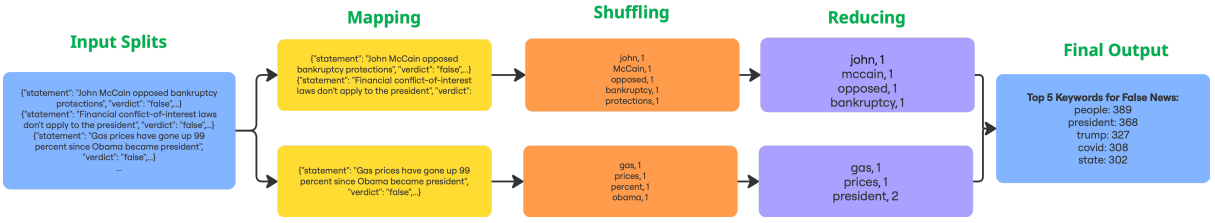
Top Originators with false/true statements



Top Months with false statements



Originators with false/true statements



Code

Pseudo Code

mapper.py

```
STOP_WORDS = {set of common English words}

def extract_month_year(date_string):
   # extract month-year from date string

def tokenize_and_filter_stop_words(text):
   # filter the line to remove the stop words and create a list of keywords, it
will remove words like a, an, the, is, and etc etc and keep keywords

for each line in standard input:
        row = json.loads(line) # parse the JSON
        # Extract the required vars from JSON

        headline = "<statement>"
        source = "<statement_source>"
        statement_originator = "<statement_originator>"
        posted_on = "<statement_date>"
        label = "<verdict>"

        # print source, label, originator, month-year

        if news is false:
                # print source, originator, month-year of news
                # Process the headline for keywords

            words = tokenize_and_filter_stop_words(headline)
            for word in words:
                    print("false_news_keyword\t", word)
```

```
        if news is true:
            # print source, originator
```

reducer.py

```
# Declare dicts to keep count of label, source, originator, source with fake news and true news, originator with fake and true news, month-year, mont h-year of fake news, keywords in fake news

for line in sys.stdin:
        # split the input on "\t"
        count each parameter from mapper and store in the dict declared abo ve

print count of label
Sort and print top 3 sources of fake news
Sort and print top 3 sources of true news
Sort and print top 3 originators of fake news
Sort and print top 3 originators of true news
Sort and print top 3 month-year with most number of fake news overall

for month_year in month_year_fake_news_count:
        Get total news for the month_year from month_year count
        Get fake news count for the month_year from month_year_fake_news_ count

        # Let's consider months-year with at least 10 news
        if total_news > 10:
            # calculate percentage of fake news

Sort and print top 3 month-year with highest percentage of fake news
Sort and print top 5 keywords in fake news
```

Functional code

The code is available in following repository

aditya-bits-cc/BDS-assignment: Assignment 1 of BDS

stop_words.json


```
{
  "stop_words": [
    "call", "upon", "still", "nevertheless", "down", "every", "forty", "’re", "always", "whole", "side",
    "n't", "now", "however", "an", "show", "least", "give", "below", "did", "sometimes", "which", "'s",
    "nowhere", "per", "hereupon", "yours", "she", "moreover", "eight", "somewhere", "within", "whereby",
    "few", "has", "so", "have", "for", "noone", "top", "were", "those", "thence", "eleven", "after", "no",
    "’ll", "others", "ourselves", "themselves", "though", "that", "nor", "just", "’s", "before", "had",
    "toward", "another", "should", "herself", "and", "these", "such", "elsewhere", "further", "next", "indeed",
    "bottom", "anyone", "his", "each", "then", "both", "became", "third", "whom", "‘ve", "mine", "take", "many",
    "anywhere", "to", "well", "thereafter", "besides", "almost", "front", "fifteen", "towards", "none", "be",
    "herein", "two", "using", "whatever", "please", "perhaps", "full", "ca", "we", "latterly", "here",
    "therefore", "us", "how", "was", "made", "the", "or", "may", "’re", "namely", "'ve", "anyway", "amongst",
    "used", "ever", "of", "there", "than", "why", "really", "whither", "in", "only", "wherein", "last", "under",
    "own", "therein", "go", "seems", "‘m", "wherever", "either", "someone", "up", "doing", "on", "rather",
    "ours", "again", "same", "over", "‘s", "latter", "during", "done", "'re", "put", "'m", "much", "neither",
    "among", "seemed", "into", "once", "my", "otherwise", "part", "everywhere", "never", "myself", "must", "will",
```

```
    "am", "can", "else", "although", "as", "beyond", "are", "too", "becomes",
"does", "a", "everyone", "but",
    "some", "regarding", "‘ll", "against", "throughout", "yourselves", "him",
"'d", "it", "himself", "whether",
    "move", "’m", "hereafter", "re", "while", "whoever", "your", "first", "amoun
t", "twelve", "serious", "other",
    "any", "off", "seeming", "four", "itself", "nothing", "beforehand", "make",
"out", "very", "already", "various",
    "until", "hers", "they", "not", "them", "where", "would", "since", "everythin
g", "at", "together", "yet", "more",
    "six", "back", "with", "thereupon", "becoming", "around", "due", "keep",
"somehow", "n‘t", "across", "all",
    "when", "i", "empty", "nine", "five", "get", "see", "been", "name", "betwee
n", "hence", "ten", "several", "from",
    "whereupon", "through", "hereby", "'ll", "alone", "something", "formerly",
"without", "above", "onto", "except",
    "enough", "become", "behind", "’d", "its", "most", "n’t", "might", "wherea
s", "anything", "if", "her", "via",
    "fifty", "is", "thereby", "twenty", "often", "whereafter", "their", "also", "any
how", "cannot", "our", "could",
    "because", "who", "beside", "by", "whence", "being", "meanwhile", "this",
"afterwards", "whenever", "mostly",
    "what", "one", "nobody", "seem", "less", "do", "‘d", "say", "thus", "unles
s", "along", "yourself", "former",
    "thru", "he", "hundred", "three", "sixty", "me", "sometime", "whose", "yo
u", "quite", "’ve", "about", "even",
    "says", "said"
  ]
}
```

mapper.py

```
#!/usr/bin/python3
import sys
import json
import re
from datetime import datetime
```

```
FALSE_NEWS_LABELS = {"FALSE", "PANTS ON FIRE"}
TRUE_NEWS_NABELS = {"TRUE", "MOSTLY-TRUE"}

with open("stop_words.json") as f:
        STOP_WORDS = set((json.load(f)["stop_words"])
# Function to extract month and year
def extract_month_year(date_str):
    date = datetime.strptime(date_str.strip(), '%m/%d/%Y')
    return date.month, date.year

def tokenize_and_filter_stop_words(text):
    # Lowercase and split text into words
    words = re.findall(r'\b[a-zA-Z]+\b(?:\'[a-zA-Z]+)?', text.lower())
    # Filter out stop words
    filtered_words = (word for word in words if word not in STOP_WORDS an
d len(word) > 1)
    return filtered_words

# Reading input from stdin
for line in sys.stdin:
    line = line.strip()
    # line = line.encode('utf-8').decode('unicode_escape')
    # Parse the JSON object
    row = json.loads(line)
    # for row in reader:
    headline = row['statement']
    source = row['statement_source'].lower()
    statement_originator = row['statement_originator'].lower()
    posted_on = row['statement_date']
    label = str(row['verdict']).upper()

    # Count the labels
    print(f"label\t{label}")

    # Count the sources
    print(f"source\t{source}")

    # Count the person who gave the statement
```

```
    print(f"statement_originator\t{statement_originator}")

    # Extract the month-year for the posted date
    month, year = extract_month_year(posted_on)
    if month and year:
        month_year = f"{month}-{year}"
        print(f"month_year\t{month}-{year}")

    if label in FALSE_NEWS_LABELS:
        print(f"fake_news_source\t{source}") # Print source if news is fake
        print(f"fake_news_statement_originator\t{statement_originator}") # Pri
nt statement_originator if news is fake
        print(f"fake_news_month_year\t{month_year}")  # Print month-year Of
fake news

        # Process the headline for keywords
        words = tokenize_and_filter_stop_words(headline)
        for word in words:
            print(f"false_news_keyword\t{word}")

    if label in TRUE_NEWS_NABELS:
        print(f"true_news_source\t{source}") # Source name with true or mostl
y-true news
        print(f"true_news_statement_originator\t{statement_originator}") # Pri
nt statement_originator if news is true or mostly-true
```

reducer.py

```
#!/usr/bin/python3
import sys

# Dicts for keeping the count
label_counts = {}
source_counts = {}
statement_originator_counts = {}
source_fake_news_count = {}
```

```
source_true_news_count = {}
statement_originator_fake_news_count = {}
statement_originator_true_news_count = {}
month_year_counts = {}
month_year_fake_news_count = {}
month_year_percentage_fake_news = {}
false_news_keyword_counts = {}

# Reading the mapper output from stdin
for line in sys.stdin:
    line = line.strip()

    key, value = line.split('\t', 1)

    # 1. Count labels
    if key == "label":
        if value in label_counts:
            label_counts[value] += 1
        else:
            label_counts[value] = 1

    # 2. Count sources
    elif key == "source":
        if value in source_counts:
            source_counts[value] += 1
        else:
            source_counts[value] = 1

    # 2. Count statement_originator
    elif key == "statement_originator":
        if value in statement_originator_counts:
            statement_originator_counts[value] += 1
        else:
            statement_originator_counts[value] = 1

    # 3. Count headlines posted per month-year
    elif key == "month_year":
        if value in month_year_counts:
```

```
            month_year_counts[value] += 1
        else:
            month_year_counts[value] = 1

    # 4. Sources with the most fake news (False, Pants on Fire)
    elif key == "fake_news_source":
        if value in source_fake_news_count:
            source_fake_news_count[value] += 1
        else:
            source_fake_news_count[value] = 1

    # 5. Sources with the most true news (True, Mostly-True)
    elif key == "true_news_source":
        if value in source_true_news_count:
            source_true_news_count[value] += 1
        else:
            source_true_news_count[value] = 1

    # statement_originator with most fake news
    elif key == "fake_news_statement_originator":
        if value in statement_originator_fake_news_count:
            statement_originator_fake_news_count[value] += 1
        else:
            statement_originator_fake_news_count[value] = 1

    # 5. statement_originator with the most true news (True, Mostly-True)
    elif key == "true_news_statement_originator":
        if value in statement_originator_true_news_count:
            statement_originator_true_news_count[value] += 1
        else:
            statement_originator_true_news_count[value] = 1

    # 6. Track fake news count per month-year
    elif key == "fake_news_month_year":
        if value in month_year_fake_news_count:
            month_year_fake_news_count[value] += 1
        else:
            month_year_fake_news_count[value] = 1
```

```
    # 7. Fake news keyword count
    elif key == "false_news_keyword":
        if value in false_news_keyword_counts:
            false_news_keyword_counts[value] += 1
        else:
            false_news_keyword_counts[value] = 1

# Count of each label e.g. TRUE, FALSE, MOSTLY-TRUE etc
print("\nVeracity count:")
for label, count in label_counts.items():
    print(f"{label} → {count}")

# Top 3 sources with most absolute false news
source_fake_news_sorted = sorted(source_fake_news_count.items(), key=l
ambda x: x[1], reverse=True)
print("\nTop 3 sources with false statements:")
for i, (source, count) in enumerate(source_fake_news_sorted[:3]):
    print(f"{source}: {count} fake news")

# Top 3 Sources with most TRUE or MOSTLY-TRUE news
source_true_news_sorted = sorted(source_true_news_count.items(), key=la
mbda x: x[1], reverse=True)
print("\nTop 3 sources with True or Mostly-true statements:")
for i, (source, count) in enumerate(source_true_news_sorted[:3]):
    print(f"{source}: {count} true news")

# Top 3 originators with most absolute false news
source_fake_news_sorted = sorted(statement_originator_fake_news_count.i
tems(), key=lambda x: x[1], reverse=True)
print("\nTop 3 originators with absolute false statements:")
for i, (source, count) in enumerate(source_fake_news_sorted[:3]):
    print(f"{source}: {count} fake news")

# Top 3 originators with most TRUE or MOSTLY-TRUE news
source_true_news_sorted = sorted(statement_originator_true_news_count.it
ems(), key=lambda x: x[1], reverse=True)
print("\nTop 3 originators with True or Mostly-true statements:")
```

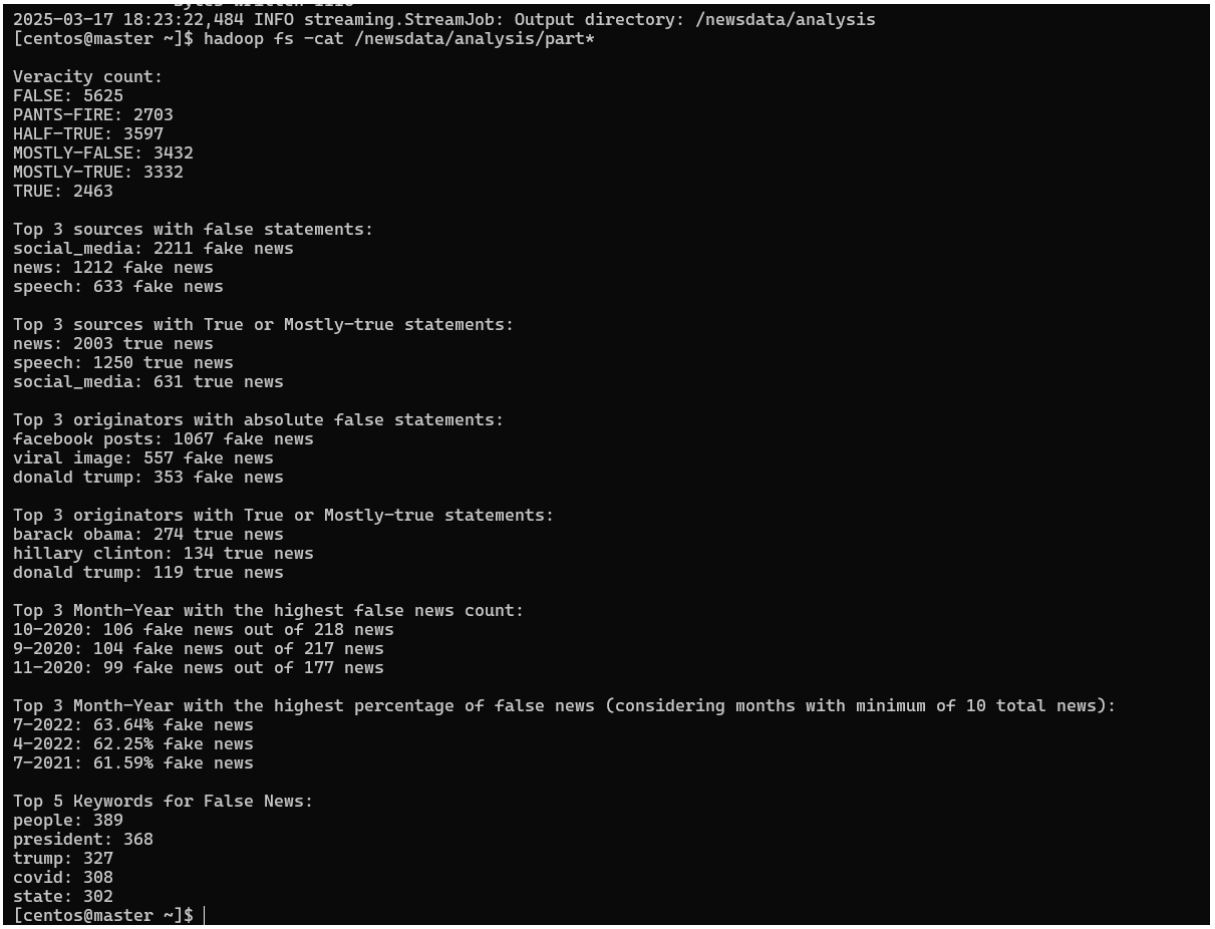
```
for i, (source, count) in enumerate(source_true_news_sorted[:3]):
    print(f"{source}: {count} true news")

# Month-year with most fake news
month_year_fake_news_sorted = sorted(month_year_fake_news_count.item
s(), key=lambda x: x[1], reverse=True)
print("\nTop 3 Month-Year with the highest overall false news count:")
for i, (month_year, count) in enumerate(month_year_fake_news_sorted[:3]):
    print(f"{month_year}: {count} fake news out of {month_year_counts.get
(month_year, 0)} news")

# Calculate the month-year with the most percentage of fake news
for month_year in month_year_fake_news_count:
    # Get the total headlines for this month-year
    total_headlines = month_year_counts.get(month_year, 0)
    fake_news_count = month_year_fake_news_count.get(month_year, 0)

    # Calculate the percentage of fake news for this month-year
    if total_headlines > 10:
        fake_news_percentage = (fake_news_count / total_headlines) * 100
        month_year_percentage_fake_news[month_year] = fake_news_percent
age

# Sort month-year by the percentage of fake news
sorted_month_year_percentage_fake_news = sorted(month_year_percentag
e_fake_news.items(), key=lambda x: x[1], reverse=True)

# month-year with the highest percentage of fake news
print("\nTop 3 Month-Year with the highest percentage of false news (consi
dering months with minimum of 10 total news):")
for month_year, percentage in sorted_month_year_percentage_fake_news[:
3]:
    print(f"{month_year}: {percentage:.2f}% fake news")

print("\nTop 5 Keywords for False News:")
sorted_keywords = sorted(false_news_keyword_counts.items(), key=lambd
a x: x[1], reverse=True)
```

```
for i, (keyword, count) in enumerate(sorted_keywords[:5]):
    print(f"{keyword}: {count}")
```

Commands executed to perform the analysis:

```
# Download the dataset
curl -L -o politifact-fact-check-dataset.zip https://www.kaggle.com/api/v1/
datasets/download/rmisra/politifact-fact-check-dataset

# Unzip and rename the file
unzip politifact-fact-check-dataset.zip
mv politifact_factcheck_data.json newsdata.json

# Put the dataset in hdfs
hadoop fs -mkdir /newsdata
hadoop fs -put newsdata.json /newsdata

# Run the job
hadoop jar /opt/hadoop-3.2.4/share/hadoop/tools/lib/hadoop-streaming-3.
2.4.jar -file stop_words.json -file mapper.py -file reducer.py -mapper "pyth
on3 mapper.py" -reducer "python3 reducer.py" -input /newsdata/newsdat
a.json -output /newsdata/analysis

# Print the analysis output
hadoop fs -cat /newsdata/analysis/*
```

Output screenshot



Output screenshot

Execution Statistics

- Number of Map tasks: 2
- Number of Reduce tasks: 1
- Memory consumption per task:
  - Peak Map Physical memory (bytes)=737529856
  - Peak Map Virtual memory (bytes)=3020111872
  - Peak Reduce Physical memory (bytes)=218513408
  - Peak Reduce Virtual memory (bytes)=4726763520
- Bytes Transferred (**Reduce shuffle bytes**)
  - Map Output Materialized Bytes: 4,444,908 bytes.
  - Reduce Shuffle Bytes: 4,444,908 bytes.

Execution statistics screenshot

```
[centos@master ~]$ hadoop jar /opt/hadoop-3.2.4/share/hadoop/tools/lib/hadoop-streaming-3.2.4.jar -file mapper.py -file reducer.py -mapper "python3 mapper.py" -reducer "python3 reducer.py"
 -input /newsdata/newsdata.json -output /newsdata/analysis
2025-03-17 18:22:59,671 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [mapper.py, reducer.py, /tmp/hadoop-unjar2408549651087025115/] [] /tmp/streamjob7742788799697734198.jar tmpDir=null
2025-03-17 18:23:00,636 INFO client.RMProxy: Connecting to ResourceManager at master/172.31.12.171:8032
2025-03-17 18:23:00,869 INFO client.RMProxy: Connecting to ResourceManager at master/172.31.12.171:8032
2025-03-17 18:23:01,075 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/centos/.staging/job_1742234537578_0006
2025-03-17 18:23:01,483 INFO mapred.FileInputFormat: Total input files to process : 1
2025-03-17 18:23:01,595 INFO mapreduce.JobSubmitter: number of splits:2
2025-03-17 18:23:01,793 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1742234537578_0006
2025-03-17 18:23:01,795 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-03-17 18:23:01,996 INFO conf.Configuration: resource-types.xml not found
2025-03-17 18:23:01,997 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-03-17 18:23:02,065 INFO impl.YarnClientImpl: Submitted application application_1742234537578_0006
2025-03-17 18:23:02,105 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1742234537578_0006/
2025-03-17 18:23:02,107 INFO mapreduce.Job: Running job: job_1742234537578_0006
2025-03-17 18:23:09,249 INFO mapreduce.Job: Job job_1742234537578_0006 running in uber mode : false
2025-03-17 18:23:09,250 INFO mapreduce.Job:  map 0% reduce 0%
2025-03-17 18:23:16,343 INFO mapreduce.Job:  map 100% reduce 0%
2025-03-17 18:23:21,383 INFO mapreduce.Job:  map 100% reduce 100%
2025-03-17 18:23:22,396 INFO mapreduce.Job: Job job_1742234537578_0006 completed successfully
2025-03-17 18:23:22,484 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=4444902
                FILE: Number of bytes written=9617738
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=9842401
                HDFS: Number of bytes written=1116
                HDFS: Number of read operations=11
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=18272
                Total time spent by all reduces in occupied slots (ms)=9861
                Total time spent by all map tasks (ms)=9136
                Total time spent by all reduce tasks (ms)=3287
                Total vcore-milliseconds taken by all map tasks=9136
                Total vcore-milliseconds taken by all reduce tasks=3287
                Total megabyte-milliseconds taken by all map tasks=18710528
                Total megabyte-milliseconds taken by all reduce tasks=10097664
```

Execution statistics 1

```
        Map-Reduce Framework
                Map input records=21152
                Map output records=164721
                Map output bytes=4115454
                Map output materialized bytes=4444908
                Input split bytes=186
                Combine input records=0
                Combine output records=0
                Reduce input groups=10
                Reduce shuffle bytes=4444908
                Reduce input records=164721
                Reduce output records=45
                Spilled Records=329442
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=182
                CPU time spent (ms)=5450
                Physical memory (bytes) snapshot=1693425664
                Virtual memory (bytes) snapshot=10764591104
                Total committed heap usage (bytes)=1646264320
                Peak Map Physical memory (bytes)=737529856
                Peak Map Virtual memory (bytes)=3020111872
                Peak Reduce Physical memory (bytes)=218513408
                Peak Reduce Virtual memory (bytes)=4726763520
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=9842215
        File Output Format Counters
                Bytes Written=1116
2025-03-17 18:23:22,484 INFO streaming.StreamJob: Output directory: /newsdata/analysis
[centos@master ~]$
```

Execution statistics 2