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

GitHub repo Reference:

https://github.com/aditya-bits-cc/BDS-assignment

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 analyze 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 Kaggle

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

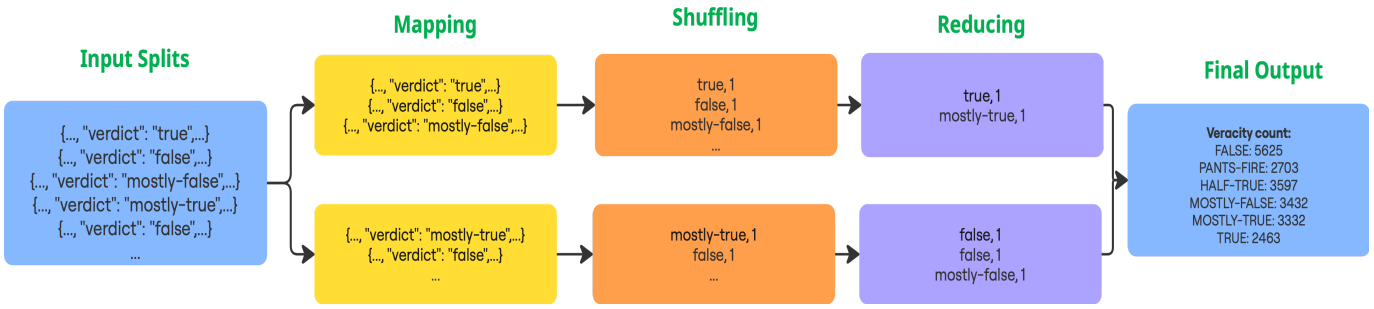
The data has been gathered from a website PolitiFact which fact checks the news. The news statements have been categorized into 6 categories: true, mostly true, half true, mostly false, false, and pants on fire

It has more than 21k news headlines fact checked

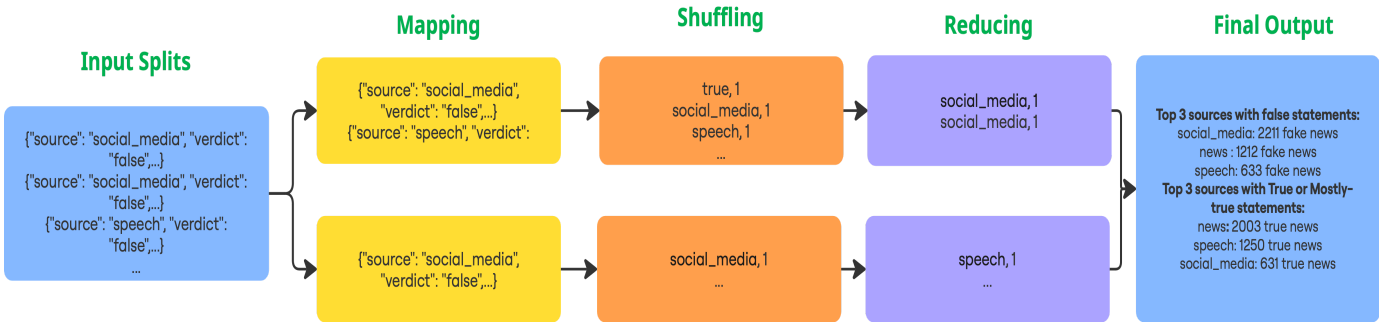
Source of dataset is rishabhmisra.github.io/publications

Map-Reduce Diagrams for each analysis task

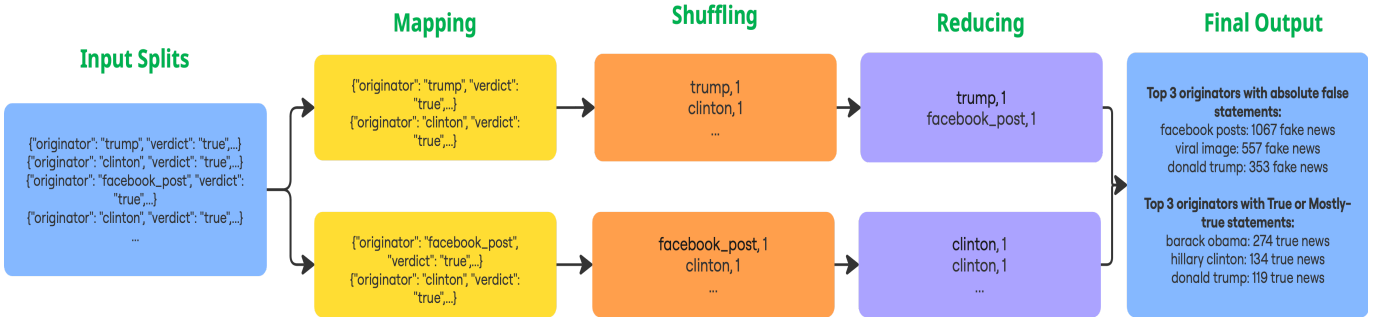
Veracity Count



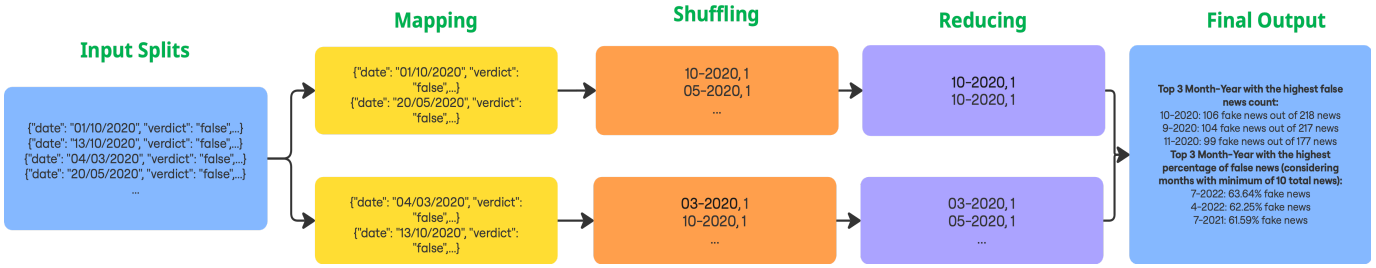
Top Sources with false/true statements



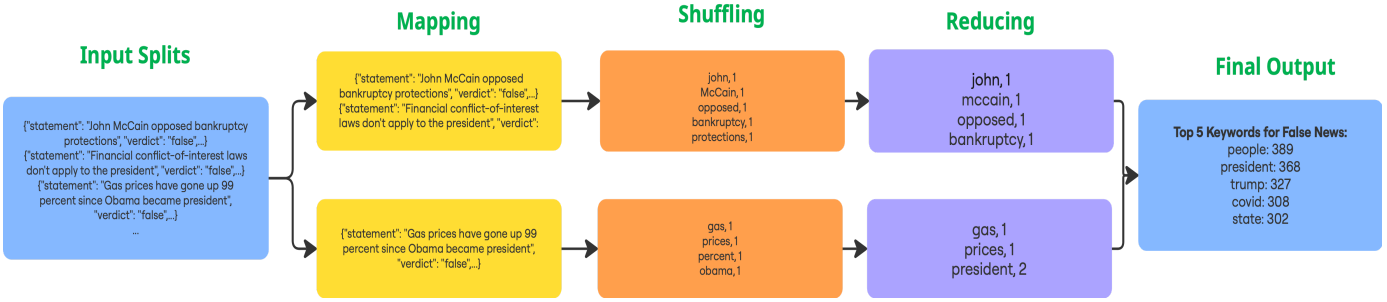
Top Originators with false/true statements



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 pushed to following repository.

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

stop_words.json


```
{
"stop_words": [
"call", "upon", "still", "nevertheless", "down", "every", "forty", "‘re", "alwa
ys", "whole", "side",
"n't", "now", "however", "an", "show", "least", "give", "below", "did", "so metimes", "which",
"'s",
"nowhere", "per", "hereupon", "yours", "she", "moreover", "eight", "some where", "within",
"whereby",
"few", "has", "so", "have", "for", "noone", "top", "were", "those", "thenc e", "eleven", "after",
"no",
"’ll", "others", "ourselves", "themselves", "though", "that", "nor", "just", "’s", "before",
"had",
"toward", "another", "should", "herself", "and", "these", "such", "elsewhe re", "further",
"next", "indeed",
"bottom", "anyone", "his", "each", "then", "both", "became", "third", "who m", "‘ve", "mine",
"take", "many",
"anywhere", "to", "well", "thereafter", "besides", "almost", "front", "fiftee n", "towards",
"none", "be",
"herein", "two", "using", "whatever", "please", "perhaps", "full", "ca", "w e", "latterly",
"here",
"therefore", "us", "how", "was", "made", "the", "or", "may", "’re", "namel y", "'ve", "anyway",
"amongst",
"used", "ever", "of", "there", "than", "why", "really", "whither", "in", "onl y", "wherein",
"last", "under",
"own", "therein", "go", "seems", "‘m", "wherever", "either", "someone", "u p", "doing", "on",
"rather",
"ours", "again", "same", "over", "‘s", "latter", "during", "done", "'re", "put", "'m", "much",
"neither",
"among", "seemed", "into", "once", "my", "otherwise", "part", "everywher e", "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

# Constants
FALSE_NEWS_LABELS = {"FALSE", "PANTS ON FIRE"}
TRUE_NEWS_LABELS = {"TRUE", "MOSTLY-TRUE"}

# Load stop words
with open("stop_words.json") as f:
    STOP_WORDS = set(json.load(f)["stop_words"])

def extract_month_year(date_str):
    """Extract month and year from a date string in MM/DD/YYYY format."""
    date = datetime.strptime(date_str.strip(), '%m/%d/%Y')
    return date.month, date.year

def tokenize_and_filter_stop_words(text):
    """Tokenize text into words, convert to lowercase, and remove stop words."""
    words = re.findall(r'\b[a-zA-Z]+\b(?:\'[a-zA-Z]+)?', text.lower())
    return (word for word in words if word not in STOP_WORDS and len(word) > 1)

# Process each line from stdin
for line in sys.stdin:
    line = line.strip()
    if not line:
        continue

    # Parse JSON line
    row = json.loads(line)

    statement = row.get('statement', '')
    source = row.get('statement_source', '').lower()
    originator = row.get('statement_originator', '').lower()
    posted_date = row.get('statement_date', '')
    label = str(row.get('verdict', '')).upper()

    # Emit general metadata
    print(f"label\t{label}")
    print(f"source\t{source}")
    print(f"statement_originator\t{originator}")

    # Extract and emit month-year info
    try:
        month, year = extract_month_year(posted_date)
        month_year = f"{month}-{year}"
        print(f"month_year\t{month_year}")
    except ValueError:
        month_year = None  # Invalid date format; skip month-year output

    # Emit info for false news
    if label in FALSE_NEWS_LABELS:
        print(f"fake_news_source\t{source}")
        print(f"fake_news_statement_originator\t{originator}")
        if month_year:
            print(f"fake_news_month_year\t{month_year}")
        for word in tokenize_and_filter_stop_words(statement):
            print(f"false_news_keyword\t{word}")

    # Emit info for true news
    if label in TRUE_NEWS_LABELS:
        print(f"true_news_source\t{source}")
        print(f"true_news_statement_originator\t{originator}")
```

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=lambda 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=lambda 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.items(), 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.items(), 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.items(), 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_percentage
```

```
# Sort month-year by the percentage of fake news
sorted_month_year_percentage_fake_news = sorted(month_year_percentage_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 (considering 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=lambda 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

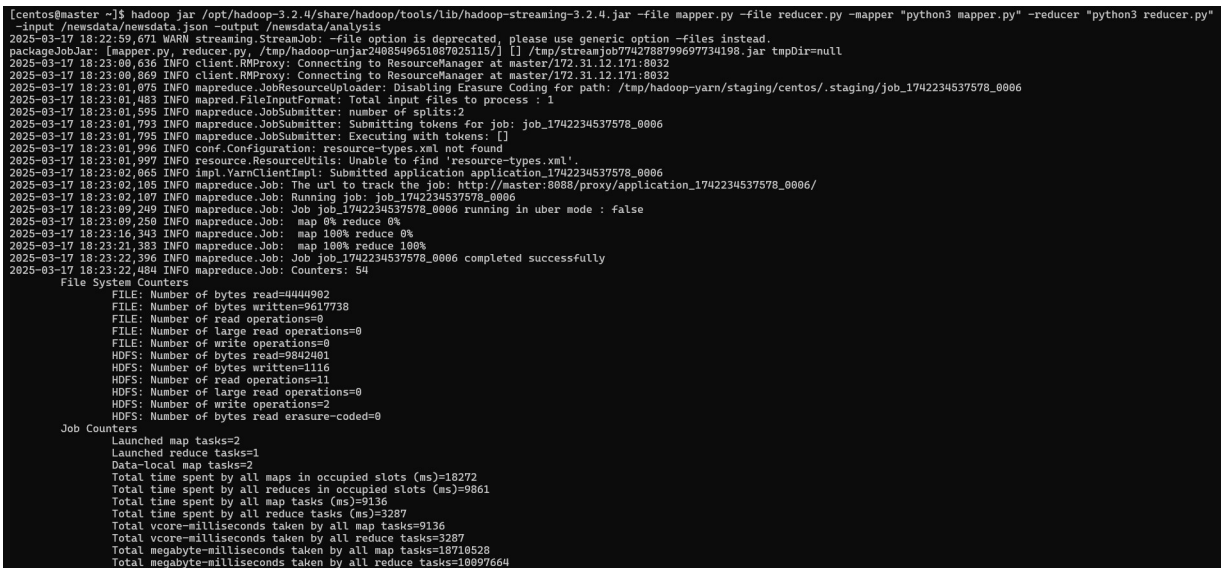
```
2025-03-17 18:23:22,484 INFO streaming.StreamJob: Output directory: /newsdata/analysis
[centos@master ~]$ hadoop fs -cat /newsdata/analysis/part*

Veracity count:
FALSE: 5625
PANTS-FIRE: 2703
HALF-TRUE: 3597
MOSTLY-FALSE: 3432
MOSTLY-TRUE: 3332
TRUE: 2463

Top 3 sources with false statements:
social_media: 2211 fake news
news: 1212 fake news
speech: 633 fake news

Top 3 sources with True or Mostly-true statements:
news: 2003 true news
speech: 1250 true news
social_media: 631 true news

Top 3 originators with absolute false statements:
facebook posts: 1067 fake news
viral image: 557 fake news
donald trump: 353 fake news

Top 3 originators with True or Mostly-true statements:
barack obama: 274 true news
hillary clinton: 134 true news
donald trump: 119 true news

Top 3 Month-Year with the highest false news count:
10-2020: 106 fake news out of 218 news
9-2020: 104 fake news out of 217 news
11-2020: 99 fake news out of 177 news

Top 3 Month-Year with the highest percentage of false news (considering months with minimum of 10 total news):
7-2022: 63.64% fake news
4-2022: 62.25% fake news
7-2021: 61.59% fake news

Top 5 Keywords for False News:
people: 389
president: 368
trump: 327
covid: 308
state: 302
[centos@master ~]$
```

Output screenshot

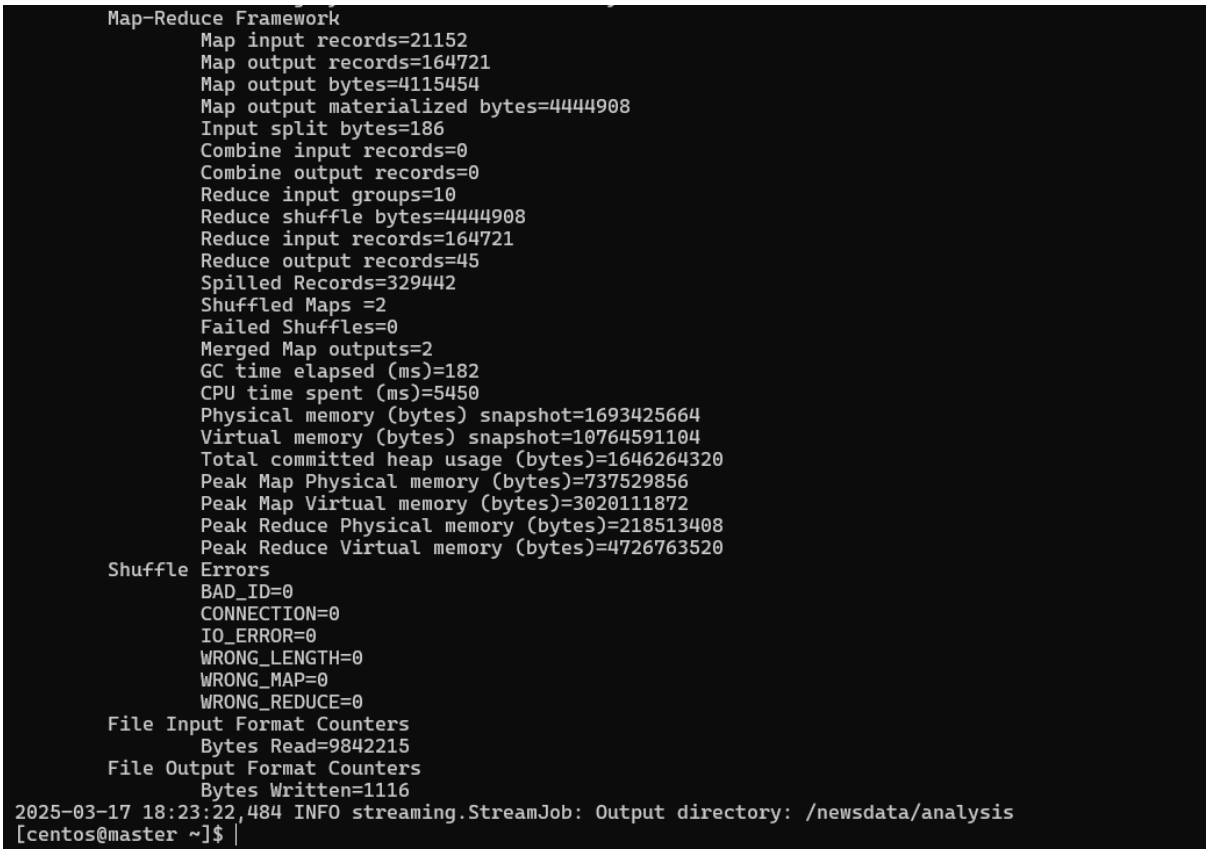
Execution Statistics

- 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



Execution statistics 1



Execution statistics 2