CS5810 (CS5710) 2023/4 High Performance Computational Infrastructures - Coursework Assignment

Student Name: Mihir Gujarathi

Student ID: 2306691

1. Introduction

In the domain of cosmetics, the products of utmost importance are cleansers, moisturizers, and sunscreens. These foundational skincare essentials are integral to daily routines and play significant roles in maintaining skin health. This report is dedicated to a comprehensive analysis of a single cosmetic dataset, with the primary aim of identifying common ingredients across cleansers, moisturizers, and sunscreens.

Cleansers, moisturizers, and sunscreens represent the cornerstone of skincare routines, offering cleansing, hydration, and sun protection, respectively. Consumers rely heavily on ingredient information to make informed purchasing decisions, while industry stakeholders use such insights to drive product innovation and marketing strategies.

1.1 Problem Description

This study addresses the challenge of analysing a cosmetic dataset to extract common ingredients among cleansers, moisturizers, and sunscreens. By identifying these ingredients, we can uncover prevailing industry trends, discern consumer preferences, and inform product development initiatives. Hence, my research question is,

What are the most prevalent ingredients shared among cleansers, moisturizers, and sunscreens, and how do these ingredient compositions vary across different skincare product categories?

1.2 Associated Dataset

This cosmetics dataset is sourced from Kaggle and can be found at Cosmetics datasets (kaggle.com). The dataset comprises information on 1472 cosmetics products found from Sephora. With a file size of 1.15MB, the dataset contains 11 columns, providing comprehensive details on various attributes of the cosmetics products. It facilitates various analyses, including ingredient trends, consumer preferences, and product suitability for different skin types.

Column	Description
Label	Type of product (e.g., cleanser, moisturizer, sunscreen)
Brand	Brand of the cosmetic product
Name	Name or identifier of the cosmetic product
Price	Price of the product in USD
Rank	Rating of the product, ranging from 1 to 5
Ingredients	List of ingredients used in the product formulation

Combination	Binary indicator representing suitability for combination skin types (0 or 1)
Dry	Binary indicator representing suitability for dry skin types (0 or 1)
Normal	Binary indicator representing suitability for normal skin types (0 or 1)
Oily	Binary indicator representing suitability for oily skin types (0 or 1)

Label	Brand	Name	Price	Rank	Ingredients	Combinat	Dry	Normal	Oily	Sensitive
Face Mask	GLAMGLO	#GLITTERM	59	4.3	Water, Polyvinyl Alcohol, Polyethylene Terephthalate, Alcohol Denat., Butylene Glycol, Glycerin, Ham	1	1	1	1	0
Treatment	KATE SOMI	+Retinol Vi	98	3.9	Water, Dimethicone, Propanediol, Polysilicone-11, Polysorbate 20, Isocetyl Stearate, Volcanic Soil, G	1	1	1	1	0
Moisturize	JOSIE MAR	100 percer	48	4.5	Organic Argania Spinosa (Argan) Kernel Oil*. *Organic. **Natural.	0	1	0	1	1
Moisturize	JOSIE MAR	100 percer	48	4.4	Argan Oil Isostearyl Esters**. **Natural.	1	1	0	1	1
Sun protec	SUPERGOO	100% Mine	38	3.7	Cyclopentasiloxane, Coco-Caprylate, Isododecane, Coco-Caprylate/Caprate, Dimethicone Crosspoly	1	1	1	1	1
Sun protec	SUPERGOO	100% Mine	34	3.8	Aloe Barbadensis Leaf Juice, Bentonite, Butyloctyl Salicylate, Calendula Officinalis Flower Extract, Ca	1	1	1	1	1
Moisturize	BIOSSANC	100% Squa	58	4.6	-100 Percent Sugarcane-Derived Squalane.	1	1	1	1	1
Treatment	DR. BRANI	2% Retinol	69	3.8	Water, Glycerin, Dicaprylyl Carbonate, Isoamyl Laurate, Cyclopentasiloxane, Butyrospermum Parkii (S	1	1	1	1	0
Treatment	PETER THO	20% Glyco	48	4.3	Water, Glycolic Acid, Potassium Hydroxide, Propanediol, Sea Water, Niacinamide, 1,2-Hexanediol, H	1	1	1	1	1
Eye cream	DR. BRANI	24/7 Retin	55	3.5	Water, Glycerin, Cyclopentasiloxane, Propanediol, Dimethiconol, Coco-Caprylate/Caprate, Hydroxye	1	1	1	1	0
Face Mask	PETER THO	24K Gold M	80	3.4	Glycerin, Water, Sodium Hyaluronate, Caffeine, Olivine Extract, Colloidal Gold, Pentylene Glycol, Carl	1	1	1	1	1
Face Mask	PETER THO	24K Gold P	75	4.1	Water, Glycerin, Carrageenan, Butylene Glycol, Ceratonia Siliqua (Carob) Gum, Colloidal Gold, Sodiu	1	1	1	1	1
Cleanser	EVE LOM	3 Muslin Cl	22	4.3	Visit the Eve Lom boutique	0	0	0	0	0
Cleanser	PETER THO	3% Glycoli	38	4.4	Water, Sodium Laureth Sulfate, Propylene Glycol, Glycolic Acid, Cocamidopropyl Betaine, Lauryl Gluc	1	1	1	1	1
Treatment	BIOEFFEC	30 Day Tre	290	5	Glycerin, Water, Sodium Hyaluronate, Tromethamine, Sodium Chloride, Hordeum Vulgare Seed Extra	1	1	1	1	1
Treatment	SEPHORA	3-in-1 Extr	23	3.9	Visit the SEPHORA COLLECTION boutique	0	0	0	0	0
Moisturize	TARTE	4-in-1 Sett	25	3.9	Water, SD Alcohol 40-B (Alcohol Denat.), Glycerin, Citrus Aurantium Dulcis (Orange) Peel Oil, Limonei	0	0	0	0	0
Face Mask	FIRST AID E	5 in 1 Boun	38	4.3	Water, Propanediol, Butylene Glycol, Cyclopentasiloxane, Glycerin, PEG-240/HDI Copolymer Bis-Dec	1	1	1	1	1
Eye cream	FIRST AID E	5 in 1 Eye C	32	3.9	Water, Glycerin, Dimethicone, C12-15 Alkyl Benzoate, Sodium Polyacrylate, Ethylhexyl Palmitate, Silic	1	1	1	1	1
Sun protec	CLARINS	50+ SPF Su	35	4.1	Visit the Clarins boutique	0	0	0	0	0
Cleanser	CLINIQUE	7 Day Scru	22	4.6	Water , Tridecyl Stearate , Tridecyl Trimellitate , Dipentaerythrityl Hexacaprylate/Hexacaprate , Butyle	0	0	0	0	0
Sun protec	SEPHORA	8 HR Mattil	23	4.2	Visit the SEPHORA COLLECTION boutique	1	1	1	1	1
Cleanser	PETER THO	8% Glycoli	40	4.3	Water, Glycolic Acid, Alcohol Denat., Methyl Gluceth-20, Propylene Glycol, Hamamelis Virginiana (Wi	1	1	1	1	1
Cleanser	PHILOSOP	A Glowing	34	4.8	Purity Made Simple Cleanser: Water, Sodium Lauroamphoacetate, Sodium Trideceth Sulfate, Limnan	1	1	1	1	1

Figure 1: Cosmetics Dataset

2. MapReduce Design and Implementation

In this section, we outline the design and implementation strategy for analysing the common ingredients among cleansers, moisturizers, and sunscreens within the cosmetic dataset using the MapReduce programming model along with Hadoop Distributed File System.

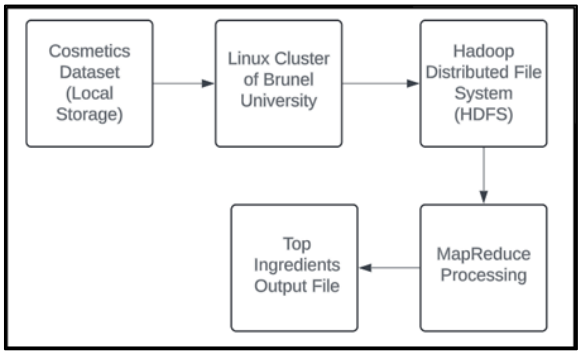

Figure 2: Overall Design

The algorithm aims to analyse a cosmetic dataset to extract common ingredients among cleansers, moisturizers, and sunscreens using the MapReduce programming paradigm. To analyse that, I will utilize the MapReduce programming model. The objective of this design is to extract ingredients from the dataset and determine their frequency of occurrence. The process will encompass three stages: HDFS input, MapReduce processing, and HDFS output.

2.1 HDFS Input

The cosmetic dataset, serving as the input for our analysis, will undergo a multi-step ingestion process to be stored in the Hadoop Distributed File System (HDFS) for efficient processing. Initially, the data will be copied from local storage to the Linux Cluster of Brunel University. From there, it will be seamlessly transferred to HDFS.

2.2 MapReduce

The mapper checks if the product is a cleanser, moisturizer, or a sunscreen by inspecting the "Label" column. If it matches, then it extracts all the ingredients from the "Ingredients" column and removes punctuations to ensure consistency and accuracy. After that, it generates key-value pairs for each non-blank ingredient, including the product type and setting the value to 1.

After receiving input from the Mapper, the Reducer aggregates ingredient counts per product type. It processes input lines, accumulates counts for each (product type, ingredient) combination, identifies the top 10 ingredients with the highest counts for each product type, and generates output showing these top ingredients along with their counts.

Following is a complete example of how the input file is being processed during the MapReduce phase.

Mapper Input

Product Type	Brand	....	Ingredients	...
Cleanser	Brand 1		Water, Glycerin, Alcohol	
Cleanser	Brand 2		Water, Alcohol	
Moisturizer	Brand 1		Water, Hyaluronic Acid, Glycerin	
Face Mask	Brand 3		Caffeine, Water	
Moisturizer	Brand 2		Glycerin, Hyaluronic Acid	
Sun Protect	Brand 1		Zinc Oxide, Glycerin, Water	
Sun Protect	Brand 4		Water	
Treatment	Brand 1		Mandelic Acid	

Above is the mapper input. Initially, the Mapper receives input data in the form of key-value pairs, with the key representing the product type and the value containing information about the product, including its ingredients. The Mapper function begins by filtering out product types that do not align with the specified goal. In the provided example, product types such as Face Mask and Treatment are excluded from further processing, while only Cleanser, Moisturizer, and Sun Protect are retained for analysis.

Following the filtering process, the Mapper proceeds to cleanse the data by removing any punctuation marks present within the ingredients list. This step ensures consistency and accuracy in ingredient representation across different products. Subsequently, for each ingredient associated with a product type, the Mapper emits a key-value pair, where the ingredient serves as the key and the product type is included as part of the value. This emitted output is labelled as Mapper Output.

Mapper Output

Key	Value
(Water, Cleanser)	1
(Glycerin, Cleanser)	1
(Alcohol, Cleanser)	1
(Water, Cleanser)	1
(Alcohol, Cleanser)	1
(Water, Moisturizer)	1
(Hyaluronic Acid, Moisturizer)	1
(Glycerin, Moisturizer)	1
(Glycerin, Moisturizer)	1
(Hyaluronic Acid, Moisturizer)	1
(Zinc Oxide, Sun Protect)	1
(Glycerin, Sun Protect)	1
(Water, Sun Protect)	1
(Water, Sun Protect)	1

Shuffle and Sort Output

Key	Value List
(Water, Cleanser)	[1, 1]
(Glycerin, Cleanser)	[1]
(Alcohol, Cleanser)	[1, 1]
(Water, Moisturizer)	[1]
(Hyaluronic Acid, Moisturizer)	[1, 1]
(Glycerin, Moisturizer)	[1, 1]
(Zinc Oxide, Sun Protect)	[1]
(Glycerin, Sun Protect)	[1]
(Water, Sun Protect)	[1, 1]

Reducer Output

Product Type	Ingredient	Total Count
Cleanser	Water	2
Cleanser	Glycerin	1
Cleanser	Alcohol	2
Moisturizer	Water	1
Moisturizer	Hyaluronic Acid	2
Moisturizer	Glycerin	2
Sun Protect	Zinc Oxide	1
Sun Protect	Glycerin	1
Sun Protect	Water	2

During the Shuffle and Sort phase, similar key-value pairs are grouped together based on their keys, facilitating the subsequent aggregation process. The Reducer function then aggregates the grouped key-value pairs by adding up the number of occurrences for each ingredient across all product types. This aggregation step enables the Reducer to determine the total count of each ingredient within the dataset.

Also, note that the actual reducer output will be displayed differently so as to beautify the results but originally it is broken down as shown in the example above.

2.3 HDFS Output

The output from the MapReduce job will be a list of the top 10 most frequently occurring ingredients for each product type (cleansers, moisturizers, and sun protectants). Each product type will have its own list of ingredients along with their respective counts.

Function	Input	Input Type	Output	Output Type
Mapper	Cosmetic dataset	Comma-separated field	Ingredient, Product Type	Map Writable
Reducer	Ingredient, Product Type	Text	Ingredient, Product Type, Count	Text

MapReduce Implementation:

The Mapper and the Reducer were coded in python and Visual Studio Code was used as Integrated Development Environment (IDE).

Mapper:

ingredients_mapper.py > ...

```python
import sys
import csv
import string

def ingredient_mapper():
    # Parse input from standard input
    reader = csv.reader(sys.stdin)
    next(reader)  # Skip header row

    # Emit (ingredient, product type, 1) for each non-blank ingredient in specified products
    for row in reader:
        product_type = row[0]
        # Check if the product is a Cleanser, Moisturizer, or Sun protect
        if "Cleanser" in product_type or "Moisturizer" in product_type or "Sun protect" in product_type:
            # Extract ingredients column
            ingredients = row[5].split(',')
            # Emit (ingredient, product type, 1) for each non-blank ingredient
            for ingredient in ingredients:
                ingredient = ingredient.strip().translate(str.maketrans('', '', string.punctuation))
                if ingredient:  # Check if ingredient is not blank
                    print(f"{ingredient}\t{product_type}\t1")

if __name__ == "__main__":
    ingredient_mapper()
```

Reducer:

ingredients_reducer.py > ...

```python
import sys
from collections import defaultdict
import heapq

def ingredient_reducer():
    current_count = defaultdict(int)

    # Parse input from standard input
    for line in sys.stdin:
        # Split the input line into ingredient, product type, and count
        parts = line.strip().split("\t")
        if len(parts) == 3:
            ingredient, product_type, count = parts
            key = (product_type, ingredient)
            current_count[key] += int(count)

    # Structure to hold the top 10 ingredients per product type
    top_ingredients = defaultdict(lambda: [])

    # Extract top 10 for each product type
    for (product_type, ingredient), count in current_count.items():
        heapq.heappush(top_ingredients[product_type], (count, ingredient))
        if len(top_ingredients[product_type]) > 10:
            heapq.heappop(top_ingredients[product_type])

    # Emit the sorted top 10 ingredients per product type
    for product_type, ingredients in top_ingredients.items():
        ingredients.sort(reverse=True)  # Sort descending by count
        print(f"\n===== Top 10 Ingredients for {product_type} =====")
        for count, ingredient in ingredients:
            print(f"{ingredient} --> {count}")
        print("\n")  # Add extra space after each product type list for clarity

if __name__ == "__main__":
    ingredient_reducer()
```

Running the MapReduce:

MapReduce necessitates a Linux environment, and fortunately, our university is equipped with a sophisticated Linux Cluster infrastructure.

a) The initial step entails logging into the Linux Cluster.

```
Microsoft Windows [Version 10.0.22631.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Mihir Gujarathi>ssh cs2306691@134.83.83.28
cs2306691@134.83.83.28's password:
Last login: Sun Apr 14 20:05:09 2024 from 134.83.252.126
(base) [cs2306691@CS5810HN ~]$
```

b) After logging in successfully, copy your dataset, mapper and reducer from local storage to Linux cluster.

```
C:\Users\Mihir Gujarathi>scp "C:\Users\Mihir Gujarathi\Downloads\HPCI\cosmet
ics.csv" cs2306691@134.83.83.28:CW
cs2306691@134.83.83.28's password:
cosmetics.csv                                 100% 1122KB  48.4KB/s   00:23

C:\Users\Mihir Gujarathi>scp "C:\Users\Mihir Gujarathi\Downloads\HPCI\ingred
ients_mapper.py" cs2306691@134.83.83.28:CW
cs2306691@134.83.83.28's password:
ingredients_mapper.py                         100% 1010    16.9KB/s   00:00

C:\Users\Mihir Gujarathi>scp "C:\Users\Mihir Gujarathi\Downloads\HPCI\ingred
ients_reducer.py" cs2306691@134.83.83.28:CW
cs2306691@134.83.83.28's password:
ingredients_reducer.py                        100% 1399    42.7KB/s   00:00
```

```
(base) [cs2306691@CS5810HN CW]$ ls
cosmetics.csv  ingredients_mapper.py  ingredients_reducer.py
(base) [cs2306691@CS5810HN CW]$ tree
.
├── cosmetics.csv
├── ingredients_mapper.py
└── ingredients_reducer.py

0 directories, 3 files
(base) [cs2306691@CS5810HN CW]$
```

c) Now, you must copy all the files into your HDFS.

```
(base) [cs2306691@CS5810HN CW]$ hadoop fs -mkdir CW
(base) [cs2306691@CS5810HN CW]$ hadoop fs -copyFromLocal * CW
(base) [cs2306691@CS5810HN CW]$ hadoop fs -ls
Found 4 items
drwxr-xr-x   - cs2306691 supergroup          0 2024-04-15 04:59 CW
drwxr-xr-x   - cs2306691 supergroup          0 2024-01-30 13:11 Lab3
drwxr-xr-x   - cs2306691 supergroup          0 2024-02-06 13:02 Lab4
drwxr-xr-x   - cs2306691 supergroup          0 2024-02-20 12:35 Lab6
(base) [cs2306691@CS5810HN CW]$ hadoop fs -ls CW
Found 3 items
-rw-r--r--   1 cs2306691 supergroup    1149414 2024-04-15 04:59 CW/cosmetics.csv
-rw-r--r--   1 cs2306691 supergroup       1010 2024-04-15 04:59 CW/ingredients_mapper.py
-rw-r--r--   1 cs2306691 supergroup       1399 2024-04-15 04:59 CW/ingredients_reducer.py
```

With this, we are all ready to perform MapReduce job in HDFS.

d) Use the Hadoop command to execute MapReduce.

```
(base) [cs2306691@CS5810HN CW]$ hadoop jar /usr/local/hadoop-3.3.6/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
    -files ingredients_mapper.py,ingredients_reducer.py  \
    -input CW/cosmetics.csv \
    -output CW/output \
    -mapper 'python ingredients_mapper.py' \
    -reducer 'python ingredients_reducer.py'
2024-04-15 05:11:33,133 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-04-15 05:11:33,239 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-04-15 05:11:33,239 INFO impl.MetricsSystemImpl: JobTracker metrics system started
```

e) Use ‘cat’ or ‘vim’ command to open your output file and see the results.

```
(base) [cs2306691@CS5810HN CW]$ hadoop fs -cat CW/output/part-00000

===== Top 10 Ingredients for Cleanser =====
Water --> 202
Glycerin --> 165
Phenoxyethanol --> 137
Butylene Glycol --> 131
Citric Acid --> 78
Fragrance --> 72
Caprylyl Glycol --> 69
Limonene --> 62
Linalool --> 61
Disodium Edta --> 56



===== Top 10 Ingredients for Moisturizer =====
Glycerin --> 204
Water --> 199
Phenoxyethanol --> 182
Butylene Glycol --> 150
Dimethicone --> 131
Sodium Hyaluronate --> 108
Tocopheryl Acetate --> 105
Caprylyl Glycol --> 101
Tocopherol --> 91
Xanthan Gum --> 83



===== Top 10 Ingredients for Sun protect =====
Phenoxyethanol --> 83
Glycerin --> 83
Water --> 80
Butylene Glycol --> 66
Dimethicone --> 65
Caprylyl Glycol --> 61
Tocopheryl Acetate --> 59
Silica --> 55
Tocopherol --> 50
Stearic Acid --> 44
```

3. Results

Glycerin appears consistently among the top ingredients across all three categories, indicating its prevalent use in skincare formulations for its moisturizing properties. Phenoxyethanol and Butylene Glycol also feature prominently across all categories, suggesting their widespread utilization as preservatives and solvent agents, respectively.

Moreover, water is a key ingredient in cleansers and moisturizers, highlighting its role as a primary solvent and hydrating agent in skincare products. Additionally, various additives such as fragrances, citric acid, and disodium EDTA are more prevalent in cleansers, reflecting their formulations tailored towards cleansing properties.

In contrast, ingredients like Dimethicone, Sodium Hyaluronate, and Xanthan Gum are more specific to moisturizers, indicating their focus on providing hydration, texture, and emollient properties to the skin.

For sun protection products, ingredients such as Silica and Stearic Acid appear, emphasizing their role in enhancing product texture and providing UV protection.

Reflection and Future Work

Reflecting on the analysis conducted, several key insights have emerged regarding the prevalent ingredients in skincare products across different categories. Understanding these trends provides valuable knowledge for both consumers and skincare product developers.

One notable observation is the consistent presence of certain ingredients, such as glycerin, phenoxyethanol, and water, underscoring their importance in skincare formulations. This highlights the significance of these ingredients in meeting consumers' expectations for efficacy and safety.

However, this analysis also raises questions regarding the potential impact of certain ingredients on skin health and the environment. Future research could explore the long-term effects of commonly used ingredients, as well as investigate alternative formulations that prioritize sustainability and skin compatibility.

Furthermore, considering the evolving preferences of consumers towards natural and organic skincare products, future work could focus on identifying and incorporating botanical ingredients with proven skincare benefits. Additionally, advancements in technology, such as nanoencapsulation and biotechnology, offer promising opportunities for enhancing the efficacy and delivery of skincare ingredients.

Appendix

The appendix includes all the files utilized in the analysis will be made available for reference and transparency purposes. This includes datasets, scripts, and output produced by MapReduce.

1. Cosmetics.csv – Dataset used

2. ingredients_mapper.py – The Mapper in Python

3. ingredients_reducer.py – The Reducer in Python

4. terminal_code – The commands used in the report

5. top_ingredients.txt – output generated by MapReduce and saved in HDFS