TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

BMCS2013 Data Engineering

ASSIGNMENT 202305

Assignment Title	:	Text Analysis on Watsons Instagram Posts
Programme	:	RDS2
Tutorial Group	:	G1,G3

Declaration

Student Name	Student ID	% Contribution	Signature	Date
TANG SHARREN	21WMR01086		sharren	17/9/23
KOONG JIE LUM	21WMR13414		koong	17/9/23
Tham Hiu Huen	21WMR13343		tham	17/9/23
Loh Eng Sin	21WMR13280		benjamin	17/9/23
		100%		

Table of Contents

1. Introduction

Our focus lies in the pharmaceutical industry, where data from Watsons Malaysia's Instagram account (@watsonsmy) serves as our primary source. This dataset has about 6000 rows and comprises 22 columns, with particular attention directed toward the "description" column, housing the textual content of each Instagram post.

To enable effective analysis, we've employed Natural Language Processing (NLP) tools. We kickstart the process by storing the dataset in Hadoop Distributed File System (HDFS) for robust storage. We then utilize Kafka for data ingestion and publication, with Kafka Consumer subscribing to process messages. Through text preprocessing, we clean and structure the data before storing it in HBase, a distributed NoSQL database.

Before diving into analytics, we enhance dataset quality by applying lemmatization and Part of Speech (POS) tagging. Our core analytical task revolves around topic modeling, aiming to unveil insights and trends within the pharmaceutical industry from Watsons Malaysia's Instagram posts. This approach provides valuable marketing and industry insights, aligning with the project's objectives.

2. Extract, Transform, Load (ETL) / Extract, Load, Transform (ELT)

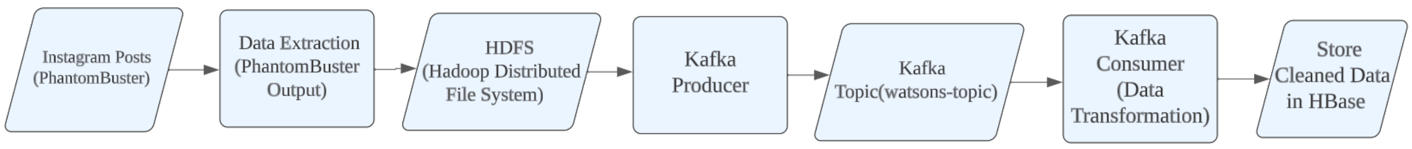
2.1. ETL/ELT Steps

Extract: We begin by scraping Watsons Malaysia's Instagram posts using the PhantomBuster website. The collected data is then securely stored in Hadoop Distributed File System (HDFS). From there, we employ a Kafka producer within the Hadoop environment to ingest and send this dataset to a Kafka topic called "watsons-topic."

Transform: Next, we leverage a Kafka consumer to subscribe to the "watsons-topic" and process the messages within it. This process involves several essential transformations, including lowercasing, URL, number, emoji, punctuation, extra whitespace, and new line character removal, handling null value ensuring the data is clean and ready for analysis.

Load: Finally, the refined dataset finds its home in HBase, a NoSQL database. For simplicity's sake, we retain the top 500 rows within HBase, providing a manageable dataset for our analytical endeavors.

Figure 1 ETL/ELT Steps and Data Storage


Instagram Posts (PhantomBuster)
Data Extraction (PhantomBuster Output)
HDFS (Hadoop Distributed File System)
Kafka Producer
Kafka Topic(watsons-topic)
Kafka Consumer (Data Transformation)
Store Cleaned Data in HBase

Link to edit

2.2. Criteria and Techniques Applied

Data Retention:

We retained only the top 500 rows in HBase for simplicity.

Data Transformation Techniques in Kafka Consumer

A function to preprocess the description column is defined in Kafka Consumer as follow:

```python
def preprocess_description(text):
    # Lowercase the text
    text = text.lower()

    # Remove numbers
    text = re.sub(r'\d+', '', text)

    # Remove links (URLs)
    text = re.sub(r'http\S+|www\S+|https\S+', '', text)

    # Remove emojis
    emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"  # Emojis in the first range
        u"\U0001F300-\U0001F5FF"  # Emojis in the second range
        u"\U0001F680-\U0001F6FF"  # Emojis in the third range
        u"\U0001F700-\U0001F77F"  # Emojis in the fourth range
        u"\U0001F780-\U0001F7FF"  # Emojis in the fifth range
        u"\U0001F800-\U0001F8FF"  # Emojis in the sixth range
        u"\U0001F900-\U0001F9FF"  # Emojis in the seventh range
        u"\U0001FA00-\U0001FA6F"  # Emojis in the eighth range
        u"\U0001FA70-\U0001FAFF"  # Emojis in the ninth range
        u"\U0001FAB0-\U0001FABF"  # Emojis in the tenth range
        u"\U0001FAC0-\U0001FAFF"  # Emojis in the eleventh range
        u"\U00002702-\U000027B0"  # Dingbats
        u"\U000024C2-\U0001F251"  # Enclosed characters
        "]+", flags=re.UNICODE)

    # Remove emojis from the text
    text = emoji_pattern.sub(r'', text)
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))

    # Remove extra whitespace
    text = ' '.join(text.split())

    # Remove newline characters (\n)
    text = text.replace('\n', ' ')

    # Remove mentions (e.g., @username)
    text = re.sub(r'@\w+', '', text)

    # Remove special characters and symbols
    text = re.sub(r'[^\w\s]', '', text)

    # Decode the text as there are text bolded using escaped Unicode
characters
    text = codecs.decode(text, 'unicode_escape')

    return text
```

2.3. Data Storage

HDFS was chosen in this project to store our data because HDFS is a distributed file system that is able to scale horizontally to even petabytes of data and this is very important for storing large amounts of data that may grow quickly from time to time. Besides that, HDFS replicates data across multiple nodes in the cluster which means that our data will still be available although some of the nodes may fail. HDFS is also a relatively cost-effective way to store large amounts of data due to the availability of deployment on commodity hardware.

HBase was chosen to store our preprocessed data because HBase is a NoSQL database designed for real-time read and write operations and this is important in storing preprocessed data for instant access. Besides that, HBase also have the advantages in the aspects of scalability and fault tolerance as we need to store and process large amounts of data from social media platforms and it is typically unstructured or semi-structured and will grow quickly in real time.

2.3.1. HBase

Hbase is used for storing the preprocessed dataset output by Kafka consumer that consumed 500 data from watsons-topic,
The HBase table is watsons-table, it has three column families: post, tagged, video

Schema of Hbase Table (watsons-table)

post:caption (String):	Description of the Instagram post caption.
post:commentCount (Integer):	The number of comments on the Instagram post.
post:description (String):	Content of the Instagram post
post:imgUrl (String):	URL of the image associated with the Instagram post.
post:isSidecar (Boolean):	Indicates whether the post is a sidecar post (multiple images or videos in a single post) or not (False).
post:likeCount (Integer):	The number of likes or reactions received on the Instagram post.
post:location (String):	Location information related to the Instagram post (if available).
post:locationId (Float):	Numeric identifier for the location (if available).
post:postUrl (String):	URL of the Instagram post on Instagram's website.
post:profileUrl (String):	URL of the user's profile who posted the Instagram content.
post:pubDate (Timestamp):	The date and time when the Instagram post was published.

post:query (String):	URL query associated with the Instagram post (if any).
post:timestamp (Timestamp):	Timestamp indicating when the data was recorded or updated.
post:type (String):	Type of Instagram post (e.g., "Video").
post:username (String):	The username of the Instagram user who posted the content.
tagged:taggedFullName1 (String):	Full name of a user tagged in the Instagram post (if any).
tagged:taggedUsername1 (String):	Username of a user tagged in the Instagram post (if any).
video:likedByViewer (Float):	Indicates whether the video was liked by the viewer (0.0 indicates not liked).
video:playCount (Float):	The number of times the video has been played.
video:videoDuration (Float):	Duration of the video content in seconds (0.0 if not a video).
video:videoUrl (String):	URL of the video associated with the Instagram post.
video:viewCount (Float):	The number of views the video has received.

<u>Example of HBase record</u>

```
 3184845383452742853              column=post:caption, timestamp=2023-09-17T05:10:03.055, value=
 3184845383452742853              column=post:commentCount, timestamp=2023-09-17T05:10:03.055, value=0
 3184845383452742853              column=post:description, timestamp=2023-09-17T05:10:03.055, value=stay healthy tv did
you know
                                   that stress can have a direct negative effect on your heart learn practical tips to
cope with
                                   stress and understand how to give your heart the best care possible with dr yee kok
meng from
                                   pantai hospital kuala lumpur pursue longevity with swisse fish oil odourless and
jampacked wi
                                  th nutrients available at watsonsmalaysia stores online chat with licensed doctors for
consult
                                  ations and eprescriptions or pharmacists for general health enquiries via watsons
virtual heal
                                  th consultation instant off rm with purchase of rm above on participating brand products
in a
                                  single receipt win a new home by lbs join malaysias largest multiexercise event ever
terms and
                                   conditions apply while stocks last stayhealthywithwatsons watsonsonzweh watsonsclub
 3184845383452742853              column=post:imgUrl, timestamp=2023-09-17T05:10:03.055,
value=https://scontent-iad3-1.cdninstag

ram.com/v/t51.2885-15/374707795_1632984377226021_2714496841855819313_n.jpg?stp=dst-jpg_e35_s10

80x1080_sh0.08&_nc_ht=scontent-iad3-1.cdninstagram.com&_nc_cat=109&_nc_ohc=N5AqKTCU1tEAX-oBTQU

&edm=AOQ1c0wBAAAA&ccb=7-5&oh=00_AfDeV4NE48TeujTY299n70wWGAFI3JcMJAbAAgHJAXMa8g&oe=64F87645&_nc
                                  _sid=8b3546
 3184845383452742853              column=post:isSidecar, timestamp=2023-09-17T05:10:03.055, value=False
 3184845383452742853              column=post:likeCount, timestamp=2023-09-17T05:10:03.055, value=11
 3184845383452742853              column=post:location, timestamp=2023-09-17T05:10:03.055, value=
 3184845383452742853              column=post:locationId, timestamp=2023-09-17T05:10:03.055, value=0.0
 3184845383452742853              column=post:postUrl, timestamp=2023-09-17T05:10:03.055,
value=https://www.instagram.com/p/Cwy2
                                  ACRPlTF/
 3184845383452742853              column=post:profileUrl, timestamp=2023-09-17T05:10:03.055,
value=https://www.instagram.com/wat
                                  sonsmy
 3184845383452742853              column=post:pubDate, timestamp=2023-09-17T05:10:03.055,
value=2023-09-05T11:07:50.000+08:00
 3184845383452742853              column=post:query, timestamp=2023-09-17T05:10:03.055,
value=https://www.instagram.com/watsonsm
                                  y/
 3184845383452742853              column=post:timestamp, timestamp=2023-09-17T05:10:03.055,
value=2023-09-05T11:13:10.576+08:00
 3184845383452742853              column=post:type, timestamp=2023-09-17T05:10:03.055, value=
 3184845383452742853              column=post:username, timestamp=2023-09-17T05:10:03.055, value=watsonsmy
 3184845383452742853              column=tagged:taggedFullName1, timestamp=2023-09-17T05:10:03.055, value=
 3184845383452742853              column=tagged:taggedUsername1, timestamp=2023-09-17T05:10:03.055, value=
 3184845383452742853              column=video:likedByViewer, timestamp=2023-09-17T05:10:03.055, value=0.0
 3184845383452742853              column=video:playCount, timestamp=2023-09-17T05:10:03.055, value=0.0
 3184845383452742853              column=video:videoDuration, timestamp=2023-09-17T05:10:03.055, value=0.0
 3184845383452742853              column=video:videoUrl, timestamp=2023-09-17T05:10:03.055,
value=https://scontent-iad3-1.cdnins

tagram.com/v/t66.30100-16/10000000_953985115665778_4667027576193653743_n.mp4?_nc_ht=scontent-i

ad3-1.cdninstagram.com&_nc_cat=110&_nc_ohc=PdNwfd_kRYEAX9R0UqZ&edm=AOQ1c0wBAAAA&ccb=7-5&oh=00_
                                  AfBI2Cw2Lmnoe_TinhL6TGXjZ2ZTyW71otuUBVdwUbuEDw&oe=64F85190&_nc_sid=8b3546
 3184845383452742853              column=video:viewCount, timestamp=2023-09-17T05:10:03.055, value=225
```

Technique

The techniques applied in Kafka consumer to store the dataframe to Hbase table are as follow:

Before inserting to a HBase table, the Hbase table must be created first in Hbase shell

Function defined in Kafka consumer to insert every row of messages consumed from watsons-topic into a Hbase table:
-It establishes a connection to the HBase
-It specifies the target HBase table name as 'watsons-table.'
-For each consumed and preprocessed record, it creates a row key using the 'postId' field.(so the resulting HBase table will only have 21 columns which is 1 column less than the source table)
-It constructs a data dictionary ('data') containing various columns and their respective values.
- It inserts this data into the HBase table using the 'put' method.
-The script prints a success message for each successfully stored row.

```
import happybase

def insert_row(row):
    try:
        #  Establishes a connection to the HBase database hosted at '10.123.51.183' on port '9090' using the 'happybase' library.        connection = happybase.Connection('10.123.51.183', port=9090)

        # specifies the target HBase table name as 'watsons-table'
        table_name = 'watsons-table'
        table = connection.table(table_name)

        # Define the data dictionary based on the provided row
        data = {
            b'post:postUrl': row.get('postUrl', '').encode('utf-8'),
            b'post:description': row.get('description', '').encode('utf-8'),
            b'post:commentCount': str(row.get('commentCount', 0)).encode('utf-8'),
            b'post:likeCount': str(row.get('likeCount', 0)).encode('utf-8'),
            b'post:pubDate': row.get('pubDate', '').encode('utf-8'),
            b'post:isSidecar': str(row.get('isSidecar', False)).encode('utf-8'),
            b'post:type': row.get('postType', '').encode('utf-8'),
            b'post:caption': row.get('caption', '').encode('utf-8'),
            b'post:profileUrl': row.get('profileUrl', '').encode('utf-8'),
            b'post:username': row.get('username', '').encode('utf-8'),
            b'tagged:taggedFullName1': row.get('taggedFullName1', '').encode('utf-8'),
            b'tagged:taggedUsername1': row.get('taggedUsername1', '').encode('utf-8'),
            b'post:imgUrl': row.get('imgUrl', '').encode('utf-8'),
            b'post:timestamp': row.get('timestamp', '').encode('utf-8'),
            b'post:query': row.get('query', '').encode('utf-8'),
            b'video:videoUrl': row.get('videoUrl', '').encode('utf-8'),
            b'video:viewCount': str(row.get('viewCount', 0.0)).encode('utf-8'),
            b'post:location': row.get('location', '').encode('utf-8'),
            b'post:locationId': str(row.get('locationId', 0.0)).encode('utf-8'),
            b'video:likedByViewer': str(row.get('likedByViewer', 0.0)).encode('utf-8'),
            b'video:videoDuration': str(row.get('videoDuration', 0.0)).encode('utf-8'),
            b'video:playCount': str(row.get('playCount', 0.0)).encode('utf-8')
        }

        # Define the row key based on the provided row
        # For each consumed and preprocessed record, it creates a row   key using the 'postId' field
        row_key = row['postId'].encode('utf-8')

        # Create a new row and insert the data into the HBase table
        table.put(row_key, data)

        print(f"Data stored successfully in row key: {row_key.decode('utf-8')}")

    except Exception as e:
        print(f"Error storing data in HBase: {str(e)}")
```

To use the defined function to store preprocessed data to Hbase:
- It consumes a specified number of records (in this case, 500) or until there are no more messages in the Kafka topic.

-For each consumed message (msg), it parses the message's value as a JSON record and assigns it to the record variable.

-The code then extracts various fields from the JSON record using record.get(field_name, default_value). For example, it extracts fields like 'postUrl,' 'description,' 'commentCount,' and more.

-Some fields, like 'description' and 'timestamp,' undergo additional processing.

-The 'description' field is preprocessed by calling the preprocess_description function, which performs text preprocessing tasks such as lowercasing, removing numbers, URLs, emojis, punctuation, and more.

-The 'postId' is split using underscores ('_') and only the first part is retained. This appears to be for formatting or simplification purposes.

-A dictionary named 'data' is created to store the extracted and preprocessed data fields. Each field name (e.g., 'postId') is mapped to its corresponding value.

-The 'data' dictionary containing the preprocessed data is passed to the insert_row function for insertion into the HBase database. This function inserts the data into the 'watsons-table' HBase table with appropriate column families and row keys.

-To avoid overwhelming the HBase database or Kafka, the script sleeps for one second (sleep(1)) after processing each message. This acts as a form of rate limiting.

```
if __name__ == '__main__':

    parsed_topic_name = 'watsons_topic'
    consumer = KafkaConsumer(parsed_topic_name, auto_offset_reset='earliest',
bootstrap_servers=['localhost:9092'], consumer_timeout_ms=2000)

    counter = 0

    for msg in consumer:
        if counter >= 500:
            break

        record = json.loads(msg.value)

        post_Url = record.get('postUrl', '')
        description = record.get('description', '')
        comment_Count = record.get('commentCount', 0)
        like_Count = record.get('likeCount', 0)
        pub_Date = record.get('pubDate', '')
        is_Sidecar = record.get('isSidecar', False)
        post_Type = record.get('type', '')
        caption = record.get('caption', '')
        profile_Url = record.get('profileUrl', '')
        username = record.get('username', '')
        tagged_FullName1 = record.get('taggedFullName1', '')
        tagged_Username1 = record.get('taggedUsername1', '')
        img_Url = record.get('imgUrl', '')
        post_Id = record.get('postId', 0.0)
        timestamp = str(record.get('timestamp', ''))
        query = record.get('query', '')
        video_Url = record.get('videoUrl', '')
        view_Count = record.get('viewCount', 0.0)
        location = record.get('location', '')
        location_Id = record.get('locationId', 0.0)
        liked_ByViewer = record.get('likedByViewer', 0.0)
        video_Duration = record.get('videoDuration', 0.0)
        play_Count = record.get('playCount', 0.0)

        description = preprocess_description(description)
        post_Id = post_Id.split('_')[0]

        data = {'postId': post_Id, 'postUrl': post_Url, 'description': description, 'commentCount':
comment_Count, 'likeCount': like_Count, 'pubDate': pub_Date, 'isSidecar': is_Sidecar, 'type': type,
'caption': caption, 'profileUrl': profile_Url, 'username': username, 'taggedFullName1':
tagged_FullName1, 'taggedUsername1': tagged_Username1, 'imgUrl': img_Url, 'timestamp': timestamp,
'query': query, 'videoUrl': video_Url, 'viewCount': view_Count, 'location': location, 'locationId':
location_Id, 'likedByViewer': liked_ByViewer, 'videoDuration': video_Duration, 'playCount':
play_Count}

        #print(data)
        insert_row(data)
        counter += 1
        sleep(1)
  if consumer is not None:
```

```
consumer.close()
```

Justification

There are several reasons why Hbase is used to store the preprocessed data:

Scalability: HBase is a NoSQL database that is built for horizontal scalability. It can efficiently handle large volumes of data, making it well-suited for storing preprocessed data generated from Instagram postings, which can be extensive and continuously growing. (Ibm.com, 2023)

HBase provides low-latency, real-time read and write capabilities. This is crucial for our use case, as we need to quickly access and update preprocessed data for analytics and other downstream tasks. (Ibm.com, 2023)

HBase allows for flexible schema design. Since Instagram postings may have varying structures, HBase accommodates this variability without the need for a fixed schema. This flexibility is essential for handling semi-structured data.

HBase supports random access to data, enabling efficient retrieval of specific records or subsets of data. This is beneficial for analytics tasks where we need to query and analyze specific aspects of the preprocessed data.

HBase maintains data consistency through automatic replication and failover mechanisms. This ensures data availability and integrity, even in the face of hardware or network failures.

HBase seamlessly integrates with other components of the Hadoop ecosystem, such as HDFS and Spark. This enables streamlined data processing workflows, as preprocessed data stored in HBase can be readily accessed and analyzed by these tools. (GeeksforGeeks, 2021)

2.3.2. Hive

Hive is used for storing the preprocessed dataset right before the text analytic task.
Hive provides a structured and tabular way to organize data, making it easier to manage and query. This structure is especially beneficial when dealing with large and complex datasets, as it enforces a schema on the data, ensuring consistency.

Hive Query Language (HQL) is SQL-like, which is familiar to many analysts and data scientists. This makes it accessible to users who are already proficient in SQL, reducing the learning curve for data analysis tasks. ((Apache.org, 2023)

Schemas of the HIVE table

post_caption (STRING)
post_comment_count (INT)
post_description (STRING)
post_img_url (STRING)
post_is_sidecar (BOOLEAN)
post_like_count (INT)
post_location (STRING)
post_location_id (FLOAT)
post_post_url (STRING)
post_profile_url (STRING)
post_pub_date (TIMESTAMP)
post_query (STRING)
post_timestamp (TIMESTAMP)
post_type (STRING)
post_username (STRING)
tagged_tagged_fullname1 (STRING)
tagged_tagged_username1 (STRING)
video_liked_by_viewer (FLOAT)
video_play_count (FLOAT)
video_video_duration (FLOAT)
video_video_url (STRING)
video_view_count (FLOAT)

Example record in Hive table (SELECT * FROM)

```
 | 0                              | its a celebration for everyone celebrate          watsons club
thversary with awesome deals including off on purchase of nd item truly onz at
watsonsmalaysi          a stores online terms and conditions apply while stocks last
promotion valid from september october wm sept          ember october em watsonsonzweh
watsonsclub kawkawdeals | NULL                              | False
| 47                                   | NULL                                 | 0.0
| NULL                                 | https://www.instagram.com/watsonsmy            |
2023-09-04T22:00:07.000+08:00            |          https://www.instagram.com/watsonsmy/
| 2023-09-05T11:13:10.576+08:00             | NULL                                   | NULL
| NULL                                   | pheiyong                                | 0.0
|
| "Photo by Watsons Malaysia on September 04        | NULL                            |  card
and           text that says 'No. 1 Retail Brand ONLINE EXCLUSIVE ELITE TUESDAY
EXTRA RM10 WORTH OF POINTS FOR EVERY RM20          0 SPEND'." | 0
| NULL                            | NU          LL                            | True
| 39.0                                 | ""                                   | 0.0
| NULL                             | https://www.instagram.com/watsonsmy                    |
NULL                             | https://www.instagram.com/watsonsmy/             |
2023-09-05T          11:13:10.576+08:00           | ""                               | watsonsmy
| NULL                                | NULL                             | 0.         0
| 0.0                                  | 0.0                                  |
| "Photo by Watsons Malaysia on September 04        | NULL                            |  hair
|  hair product and text."                       | NULL                            | NULL
| https://scontent-iad3-2.cdninstagram.com
/v/t39.30808-6/372074815_870086487806343_6882300926302284928_n.jpg?stp=dst-jpg_
e15_fr_s1080x1080&_nc_ht=sco
ntent-iad3-2.cdninstagram.com&_nc_cat=103&_nc_ohc=sp0lH8n5i_YAX-7VIC9&edm=
AOQ1c0wAAAAA&ccb=7-5&oh=00_AfDlWv
K-cHEjhRyi-sJBueETiuDruQN0DvTWYKP0DW7_QA&oe=64FBC4C7&_nc_sid=8b354
6 | NULL                                | 17                                 | ""
| NULL                             | https://www.instagram.com/p/Cwynj3yp0KJ/          | NULL
| 2023-09-05T09:00:06.000+08:00                  | https://www.instagram.com/watso
nsmy/      | 2023-09-05T11:13:10.576+08:00              | ""                              |
NULL                                | NULL                            | NULL
| 0.0                                  | 0.0                                  |
| ""                                   | 0                                | stay healt          hy tv did you
know that stress can have a direct negative effect on your heart learn practical tips to cope
with stress and understand how to give your heart the best care possible with dr yee kok
meng from pantai           hospital kuala lumpur pursue longevity with swisse fish oil
odourless and jampacked with nutrients availabl          e at watsonsmalaysia stores online
chat with licensed doctors for consultations and eprescriptions or pharm          acists for
general health enquiries via watsons virtual health consultation instant off rm with
purchase of           rm above on participating brand products in a single receipt win a new
home by lbs join malaysias largest          multiexercise event ever terms and conditions
apply while stocks last stayhealthywithwatsons watsonsonzweh          watsonsclub |
https://scontent-iad3-1.cdninstagram.com/v/t51.2885-15/374707795_1632984377226021_
27144968418
55819313_n.jpg?stp=dst-jpg_e35_s1080x1080_sh0.08&_nc_ht=scontent-iad3-1.cdninstagr
am.com&_nc_cat=109&_nc_oh
c=N5AqKTCU1tEAX-oBTQU&edm=AOQ1c0wBAAAA&ccb=7-5&oh=00_AfDeV4NE
48TeujTY299n70wWGAFI3JcMJAbAAgHJAXMa8g&oe=64F8
7645&_nc_sid=8b3546 | false                    | 11                    | ""
| 0.0                    | https://www.instagram      .com/p/Cwy2ACRPlTF/        |
https://www.instagram.com/watsonsmy           | NULL                         |
https://www.instagram.com/watsonsmy/          | NULL                    | "       "
| watsonsmy                   | ""                                    | ""
| 0.0                                | 0.0                        | 0.0                        |
ht
tps://scontent-iad3-1.cdninstagram.com/v/t66.30100-16/10000000_953985115665778_466
7027576193653743_n.mp4?_n
c_ht=scontent-iad3-1.cdninstagram.com&_nc_cat=110&_nc_ohc=PdNwfd_kRYEAX9R0
UqZ&edm=AOQ1c0wBAAAA&ccb=7-5&oh=0
0_AfBI2Cw2Lmnoe_TinhL6TGXjZ2ZTyW71otuUBVdwUbuEDw&oe=64F85190&_nc_s
id=8b3546 | 225.0                          |
+-------------------------------------------------+--------------------------------------+-----------
-------------------------------------+-------------------------------------------------+-----------
-------------------------+-------------------------------------+--------------------------------------
-----------+-------------------------------------+---------------------------------------------------+--
-----------------------------------------------+-----------------------------------+------------------
--------------------------------+-------------------------------------+-----------------------------------
-----------------+----------------------------------------+--------------------------------------------
-+-------------------------------------------+-----------------------------------------+------------
-------------------------+-----------------------------------------+-----------------------------------
----------------+--------------------------------------+
500 rows selected (2.328 seconds)
```

Technique used to save preprocessed dataset to HIVE

In beeline

Step 1: Create and use database(watsonsdb) in hive to store the table (watsonsNlpProcessed)

```
CREATE DATABASE watsonsdb;
USE watsonsdb;
```

Step 2: Create table in HIVE

Since the csv file stored in HDFS have a header(which is the first row) ,so we will skip the first row when creating the table

```
CREATE TABLE watsonsNlpProcessed(
   post_caption STRING,
   post_comment_count INT,
   post_description STRING,
   post_img_url STRING,
   post_is_sidecar BOOLEAN,
   post_like_count INT,
   post_location STRING,
   post_location_id FLOAT,
   post_post_url STRING,
   post_profile_url STRING,
   post_pub_date TIMESTAMP,
   post_query STRING,
   post_timestamp TIMESTAMP,
   post_type STRING,
   post_username STRING,
   tagged_tagged_fullname1 STRING,
   tagged_tagged_username1 STRING,
   video_liked_by_viewer FLOAT,
   video_play_count FLOAT,
   video_video_duration FLOAT,
   video_video_url STRING,
   video_view_count FLOAT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
TBLPROPERTIES (
   "skip.header.line.count"="1"
);
```

Step 2: Loading the CSV file in hdfs into hive

```
LOAD DATA INPATH
'hdfs://10.123.51.183/watsons-nlp-processed-folder/watsons-nlp-proces
sed.csv' INTO TABLE watsonsNlpProcessed;
```

2.3.3. CSV

A.watsons-dataset.csv

This is used for storing the original dataset crawled from the Instagram @watsonsmy

Schema of watsons-dataset.csv

postUrl STRING

description STRING

commentCount STRING

likeCount STRING

pubDate STRING

isSidecar STRING

type STRING

caption STRING

profileUrl STRING

username STRING

taggedFullName1 STRING

taggedUsername1 STRING

imgUrl STRING

postId STRING

timestamp STRING

query STRING

videoUrl STRING

viewCount STRING

location STRING

locationId STRING

likedByViewer STRING

videoDuration STRING

playCount STRING

<u>Example record in watsons-dataset.csv</u>

```
Row(_c0='https://www.instagram.com/p/CKU_rqhJWcz/', _c1='What is going on here? 🤔 Stay tuned to our
page and catch them in action VERY soon! 😍', _c2=None, _c3=None, _c4=None, _c5=None, _c6=None,
_c7=None, _c8=None, _c9=None, _c10=None, _c11=None, _c12=None, _c13=None, _c14=None,
_c15=None, _c16=None, _c17=None, _c18=None, _c19=None, _c20=None, _c21=None, _c22=None),
 Row(_c0='#WatsonsMalaysia #HappyBeautifulYear #ComingSoon #StayTuned"', _c1='78', _c2='1057',
_c3='2021-01-22T01:00:46.000Z', _c4='FALSE', _c5='Video', _c6=None,
_c7='https://www.instagram.com/watsonsmy', _c8='watsonsmy', _c9='Summer Grace',
_c10='mysummergrace',
_c11='https://scontent-iad3-2.cdninstagram.com/v/t51.2885-15/140227005_746417939311299_47017630132
11894985_n.jpg?stp=dst-jpg_e15&_nc_ht=scontent-iad3-2.cdninstagram.com&_nc_cat=103&_nc_ohc=1EU
Fhxj9WS4AX80lTHy&edm=ABmJApABAAAA&ccb=7-5&ig_cache_key=MjQ5MTg5NjU3MTM2MzY4
MjA5OQ%3D%3D.2-ccb7-5&oh=00_AfC0mJoZDo4EJj1PLcTevZ_BS0snMD7RfhlkQ31K3uIZjw&oe=64F
B954F&_nc_sid=b41fef', _c12='2491896571363682099_1508019062', _c13='2023-09-05T03:23:19.968Z',
_c14='https://www.instagram.com/watsonsmy/',
_c15='https://scontent-iad3-1.cdninstagram.com/v/t50.2886-16/139693650_2964836953747135_2343841865
70693386_n.mp4?efg=eyJ2ZW5jb2RlX3RhZyI6InZ0c192b2RfdXJsZ2VuLjk2MC5mZWVkLmRlZmF1bH
QiLCJxZV9ncm91cHMiOiJbXCJpZ193ZWJfZGVsaXZlcnlfdnRzX290ZlwiXSJ9&_nc_ht=scontent-iad3-1.c
dninstagram.com&_nc_cat=107&_nc_ohc=MPbl9wAoP0AAX8fctmG&edm=ABmJApABAAAA&vs=1789
8626587748554_2745060715&_nc_vs=HBksFQAYJEdGS09Vd2ktN25IbGdJZ0tBQXJiZ3dVdHMwQURia
1lMQUFBRhUAAsgBABUAGCRHSWszWHdqM043V2dESDBCQU1fbmZRekZiRG9EYmtZTEFBQUY
VAgLIAQAoABgAGwGIB3VzZV9vaWwBMBUAACaU8%2FPNl63LPxUCKAJDMywXQCaIMSbpeNU
YEmRhc2hfYmFzZWxpbmVfMV92MREAdeoHAA%3D%3D&ccb=7-5&oh=00_AfDeIRo2Nd5R1hPktLC
nv4UbdovDHsMZiG3fxgRjyiYutA&oe=64F7F559&_nc_sid=b41fef', _c16='54528', _c17=None,
_c18=None, _c19='FALSE', _c20='11.266', _c21=None, _c22=None)
```

B.watsons-nlp-processed.csv

This is used for storing the preprocessed dataset right before the text analytic task (topic modeling). This file is then saved to HDFS

<u>Schema of CSV Table</u>

post:caption - STRING

post:commentCount - STRING

post:description - STRING

post:imgUrl - STRING

post:isSidecar - STRING

post:likeCount - STRING

post:location - STRING

post:locationId - STRING

post:postUrl - STRING

post:profileUrl - STRING

post:pubDate - STRING

post:query - STRING

post:timestamp - STRING

post:type - STRING

post:username - STRING

tagged:taggedFullName1 - STRING

tagged:taggedUsername1 - STRING

video:likedByViewer - FLOAT

video:playCount - FLOAT

video:videoDuration - FLOAT

video:videoUrl - STRING

video:viewCount - FLOAT

Example of CSV records with its header

```
Row(_c0='post:caption', _c1='post:commentCount', _c2='post:description',
_c3='post:imgUrl', _c4='post:isSidecar', _c5='post:likeCount',
_c6='post:location', _c7='post:locationId', _c8='post:postUrl',
_c9='post:profileUrl', _c10='post:pubDate', _c11='post:query',
_c12='post:timestamp', _c13='post:type', _c14='post:username',
_c15='tagged:taggedFullName1', _c16='tagged:taggedUsername1',
_c17='video:likedByViewer', _c18='video:playCount',
_c19='video:videoDuration', _c20='video:videoUrl', _c21='video:viewCount'),
Row(_c0=None, _c1='0', _c2='lets work together for a trashfree future its
all about simple actions that make a big impact and thats not all with
every purchase of naturals by watsons cherry blossom range rm will be
donated to tcs turtle conservation society to support their vital work in
protecting turtles and their habitats join us in making a lasting
difference for our beautiful planet win barbie themed staycation with grand
hyatt terms and conditions apply while stocks last watsonsgogreen
gogreenwithwatsons greenisthenewbeautiful',
_c3='https://scontent-iad3-1.cdninstagram.com/v/t39.30808-6/359808505_84389
1897092469_413699465768956374_n.jpg?stp=dst-jpg_e15&_nc_ht=scontent-iad3-1.
cdninstagram.com&_nc_cat=109&_nc_ohc=OtS_UezA7G8AX8lnnYN&edm=ABmJApAAAAAA&c
cb=7-5&ig_cache_key=MzE0NjY0MDUzNDc0MjIyODIyMw%3D%3D.2-ccb7-5&oh=00_AfA0u9D
lsRyl3aDlTp00m4uMWBAEBSgsOAYlIp702gYxLQ&oe=64FBBF2D&_nc_sid=b41fef',
_c4='True', _c5='66', _c6=None, _c7='0.0',
_c8='https://www.instagram.com/p/CurHOemJChL/',
_c9='https://www.instagram.com/watsonsmy',
_c10='2023-07-14T18:00:05.000+08:00',
_c11='https://www.instagram.com/watsonsmy/',
_c12='2023-09-05T11:13:56.218+08:00', _c13=None, _c14='watsonsmy',
_c15='TCS Malaysia', _c16='tcsmalaysia', _c17='False', _c18='0.0',
_c19='0.0', _c20=None, _c21='0.0')
```

Techniques used to handle CSV files in HDFS

To save the preprocessed dataframe (hbase_df) as CSV in HDFS after lemmatization with pos and contraction handling

```
import findspark
import os
os.environ["SPARK_HOME"]='/home/user2/spark'

findspark.init()

from pyspark.sql import SparkSession
sparkSession =
SparkSession.builder.appName("example-pyspark-read-and-write").getOrC
reate()
data = hbase_df
df = sparkSession.createDataFrame(data)
# df.write.csv("hdfs://10.123.51.183/watsons-nlp-processed.csv")
output_hdfs_path =
"hdfs://10.123.51.183/watsons-nlp-processed-folder"
df.repartition(1).write.mode("overwrite").format("csv").option("heade
r", "true").save(output_hdfs_path)
```

To rename the CSV generated by repartition to a consistent name
so that we still can access it after rerun the repartition command (as it will generate csv with a different file name everytime you run it)
In this way, we can access the NLP preprocessed csv file
with hdfs path name =
“hdfs://10.123.51.183/watsons-nlp-processed-folder/watsons-nlp-processed.csv”

```
from hdfs import InsecureClient

# HDFS Configuration
hdfs_config = {
    "host": "10.123.51.183",
    "port": 50070,  # Default HDFS port
    "user": "tarcbda"
}

# Create an insecure HDFS client
```

```
hdfs_client =
InsecureClient(f"http://{hdfs_config['host']}:{hdfs_config['port']}",
user=hdfs_config['user'])

# Specify the HDFS folder path and the prefix of the CSV file to
rename
hdfs_folder_path = "/watsons-nlp-processed-folder"
csv_file_prefix = "part-00000"

# List files in the folder
file_list = hdfs_client.list(hdfs_folder_path)

# Print the list of files in the folder
print("Files in the folder:")
for file_name in file_list:
    print(file_name)

# Find the CSV file to rename
csv_file_to_rename = None
for file_name in file_list:
    if file_name.startswith(csv_file_prefix):
        csv_file_to_rename = file_name
        break

if csv_file_to_rename:
    try:
        # Perform the rename operation
        source_path = f"{hdfs_folder_path}/{csv_file_to_rename}"
        destination_path =
f"{hdfs_folder_path}/watsons-nlp-processed.csv"
        hdfs_client.rename(source_path, destination_path)
        print("CSV file renamed successfully.")
    except Exception as e:
        print(f"Error renaming CSV file: {str(e)}")
else:
    print("CSV file with the specified prefix not found in the
folder.")
```

To retrieve the CSV stored in HDFS

```
import findspark
import os
os.environ["SPARK_HOME"]='/home/user3/TangSharren/spark'
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .appName("Assignment") \
    .getOrCreate()

# Define the HDFS path to the file
hdfs_file_path =
"hdfs://10.123.51.183/watsons-nlp-processed-folder/watsons-nlp-proces
sed.csv"

# Read the csv file into a dataframe named as df
df = spark.read.csv(hdfs_file_path, header = True)
```

3. Selected NLP Tool: Lemmatizer & Contraction Handler with POS tagger

As the preprocessed dataset by Kafka consumer is stored in HBase table, we have to retrieve the HBase table to perform further preprocessing using the NLP tool: Lemmatizer & Contraction Handler with POS tagger.

Lemmatizer with POS Tagger

Lemmatization with part-of-speech (POS) tagging is an important preprocessing step for topic modeling on Instagram posts from Watsons Malaysia

Instagram posts from a pharmacy can contain a wide range of content, including product descriptions, customer reviews, promotional messages, and more. Lemmatization with POS tagging helps in understanding the context in which words are used. (GeeksforGeeks, 2019) This context is crucial for topic modeling because it allows you to distinguish between topics related to products, customer experiences, marketing campaigns, and other relevant themes.

Also, Instagram posts often contain noise in the form of slang, abbreviations, and variations in word forms (e.g., "buying" vs. "buy" "bought"). Lemmatization reduces this noise by converting words to their base or dictionary forms. This simplification makes it easier to identify and extract meaningful topics from the text data.

Lemmatization normalizes the text data, ensuring that different forms of the same word are treated as a single word. This is crucial for topic modeling, as it helps consolidate terms related to the same topic. For example, "buy," "buying," and "bought" can all be lemmatized to "buy," making it clear that they refer to the same action.

Lemmatization improves topic coherence by grouping words with similar meanings together. When performing topic modeling, words are often grouped into topics based on their co-occurrence patterns. Lemmatization ensures that synonymous words are treated as the same word, enhancing the interpretability and coherence of the resulting topics.

Lemmatized text data can lead to better topic modeling results. When words are in their base forms, topic modeling algorithms can more effectively identify patterns and associations between words, leading to more accurate topic assignments.

After topic modeling, we have to label topics with meaningful terms. Lemmatized text provides a cleaner and more intuitive set of words for labeling topics, making it easier for analysts to understand and interpret the topics.

Lemmatization enhances the interpretability of topics. Since words are in their base forms, the topics are more likely to contain common and easily understandable words, facilitating human interpretation.

Lemmatization can help reduce the dimensionality of the text data, making it more manageable for topic modeling algorithms. This can lead to faster model training and improved performance.

Contraction Handler

A contraction handler is a text preprocessing technique that expands contractions (e.g., "can't" to "cannot") to improve text tokenization, standardization, and overall data quality for natural language processing tasks. (GeeksforGeeks, 2020)

This tool helps maintain text integrity. This is because Instagram posts often contain informal language, which includes contractions (e.g., "can't" for "cannot," "won't" for "will not"). Handling contractions ensures that the original text's integrity is preserved. Without proper handling, contractions might be split into two words during tokenization, potentially leading to fragmented or less meaningful topics.

Also, when contractions are handled correctly, tokenization becomes more accurate. This is important for topic modeling because the quality of tokens impacts the topics' coherence and interpretability.

As the contraction can also be considered as noise in the text data. By expanding contractions, we simplify the text and remove unnecessary variations. This reduction in noise makes it easier for topic modeling algorithms to identify and extract meaningful topics from the text.

By using this tool, it standardizes the Instagram text data. It ensures that different forms of the same word (contracted and uncontracted) are treated consistently. This standardization simplifies the topic modeling process by reducing vocabulary size and making it easier to identify related words.

Handling contractions can lead to improved topic coherence. When related words are consistently represented, topic modeling algorithms can identify patterns and associations more effectively, resulting in topics that are more coherent and interpretable. Therefore, this led to better topic modeling results

Handling contractions enhances the interpretability of topics. Since words are presented in their expanded forms, topics are more likely to contain common and easily understandable words, making it easier for analysts to understand and interpret the topics.

Reduced Ambiguity: Some contractions can be ambiguous (e.g., "he'll" can mean "he will" or "he shall"). By expanding contractions, you reduce this ambiguity, ensuring that topic modeling algorithms can make more precise associations between words.

Techniques used

First step:

To retrieve all rows of data from an HBase table named 'watsons-table':
- For each row retrieved from HBase, the code decodes the byte-encoded values. HBase stores data as bytes, so the code converts the byte-encoded key-value pairs into a Python dictionary where both keys and values are decoded from bytes to UTF-8 strings. This dictionary, decoded_value, represents a single row of data.

-The decoded_value dictionary for each row is appended to the data list, which is used to collect all rows of data.

-After iterating through all rows in the HBase table, the fetch_hbase_data function returns the collected data as a list of dictionaries, where each dictionary represents a row of data with UTF-8 string keys and values.

-All rows retrieved are in the ‘hbase_data’ variable

```python
import happybase


connection = happybase.Connection('10.123.51.183', port=9090)
table_name = 'watsons-table'
table = connection.table(table_name)


# Fetch all rows from HBase
def fetch_hbase_data():
    data = []
    for key, value in table.scan():
        # Decode the byte-encoded values from HBase
        decoded_value = {k.decode('utf-8'): v.decode('utf-8') for k, v in value.items()}
        data.append(decoded_value)
    return data
```

```
hbase_data = fetch_hbase_data()
hbase_data
```

Second step

All rows in the HBase table are now loaded into the variable hbase_data, lemmatization with POS tagger& contraction handler will be applied to the hbase_data

-Expanding Contractions: It uses the contractions.fix() function to expand contractions like "don't" to "do not."

-Tokenization: The text is tokenized into individual words using NLTK's word_tokenize function.

-Stopwords Removal: Common stopwords are removed from the tokenized words.

-Part-of-Speech (POS)Tagging: NLTK's pos_tag function is used to perform part-of-speech tagging on the remaining words.

-Lemmatization: A function called get_wordnet_pos is defined to map part-of-speech tags to WordNet tags (e.g., 'J' for adjective, 'V' for verb). This mapping is used to lemmatize words based on their POS tags using the WordNet lemmatizer. The lemmatized words are joined back into a sentence, and the preprocessed text is returned.

^As the HBase table have 3 column-family: post,tagged and video. We have to access a column through its column-family name also.

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk import pos_tag
from nltk.stem import WordNetLemmatizer
import contractions


nltk.download('punkt')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')

# Initialize the stopwords list
stop_words = set(stopwords.words('english'))

lemmatizer = WordNetLemmatizer()

# Define a function to preprocess a single text entry
def preprocess_text(text):
    # Expand contractions
    text = contractions.fix(text)

    # Tokenize the text
    words = word_tokenize(text)

    # Remove stopwords
    filtered_words = [word for word in words if word.lower() not in stop_words]

    # Perform POS tagging
    tags = pos_tag(filtered_words)

    # Define a function to map POS tags to WordNet tags for lemmatization
    def get_wordnet_pos(tag):
        if tag.startswith('J'):
            return nltk.corpus.wordnet.ADJ
        elif tag.startswith('V'):
            return nltk.corpus.wordnet.VERB
        elif tag.startswith('N'):
            return nltk.corpus.wordnet.NOUN
        elif tag.startswith('R'):
            return nltk.corpus.wordnet.ADV
        else:
            return nltk.corpus.wordnet.NOUN  # Default to noun if not found

    # Lemmatize words based on their POS tags
    lemmatized_words = [lemmatizer.lemmatize(word, get_wordnet_pos(tag)) for word, tag
in tags]
```

```
    # Join the lemmatized words back into a sentence
    lemmatized_text = ' '.join(lemmatized_words)

    return lemmatized_text

# Iterate through your HBase data and preprocess the "post:description" column
for row in hbase_data:
    if 'post:description' in row:
        description = row['post:description']
        processed_text = preprocess_text(description)
        row['post:description'] = processed_text
```

4. Selected Text Analytics Tasks

Topic modeling

Topic modeling is a natural language processing (NLP) technique used for automatically identifying topics or themes within a collection of text documents. It is a valuable tool for tasks such as document clustering, text summarization, content recommendation, and information retrieval. (MonkeyLearn Blog, 2019)

Performing topic modeling on Watson's Instagram posts can provide valuable insights into the subjects, trends, and discussions that are relevant to the pharmacy's audience. This analysis can help the pharmacy understand customer interests, concerns, and preferences, allowing them to tailor their content, products, and services more effectively. Additionally, topic modeling can identify emerging health and wellness topics, enabling Watsons to stay up-to-date with industry trends and consumer needs.

TF-IDF

Term Frequency - Inverse Document Frequency (TF-IDF) is a widely used statistical method in natural language processing and information retrieval. It measures how important a term is within a document relative to a collection of documents (i.e., relative to a corpus). Words within a text document are transformed into importance numbers by a text vectorization process. There are many different text vectorization scoring schemes, with TF-IDF being one of the most common. (Fatih Karabiber, 2023)

TF-IDF can identify and highlight important keywords or terms within Watson's posts. By calculating the TF-IDF scores for each term, words are relatively unique and significant in individual posts or across all posts can be identify. Analyzing the changing TF-IDF scores of specific terms over time can help detect trends in Watson's posts. For example, it can reveal which topics are gaining or losing popularity among the audience. TF-IDF can help identify the most influential posts by considering the terms that have the highest TF-IDF scores. These posts may have sparked engagement, discussions, or interactions, making them valuable for analysis.

Technique used

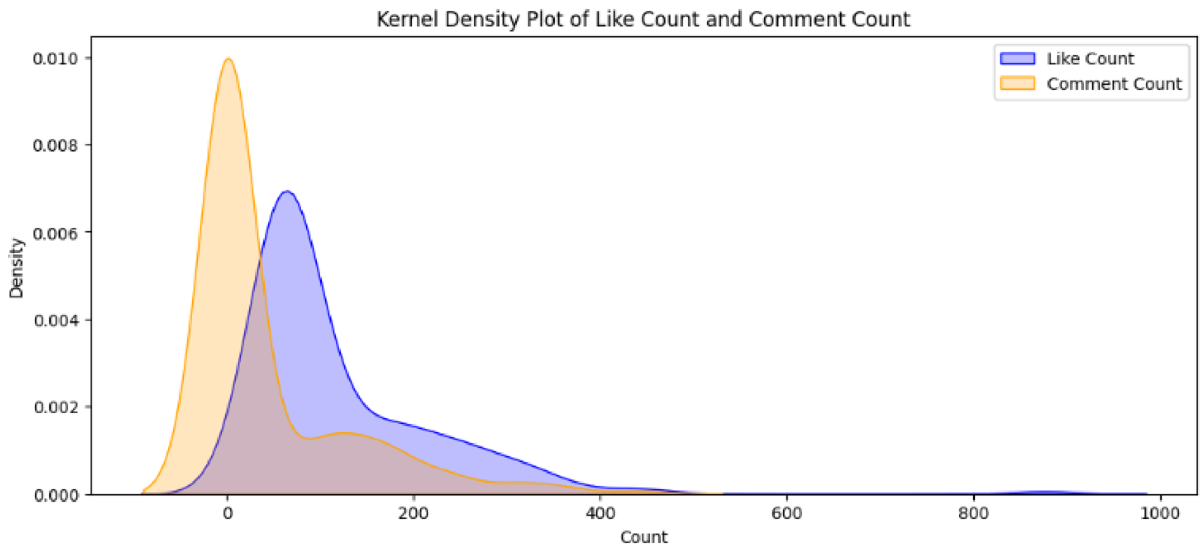
The provided code initializes a Spark session and reads a CSV file located in HDFS. This approach leverages Spark's distributed computing capabilities for efficient data processing and analysis. Once the data is loaded into a Spark DataFrame, it can be easily manipulated and analyzed.

```
# Initialize Spark session
spark = SparkSession.builder.appName("Task4").getOrCreate()

#Assign hdfs path
hdfs_file_path =
"hdfs://10.123.51.183/watsons-nlp-processed-folder/watsons-nlp-processed.csv"

#Read csv file on Spark
df = spark.read.csv(hdfs_file_path)
```

<u>Visualization of analysis</u>

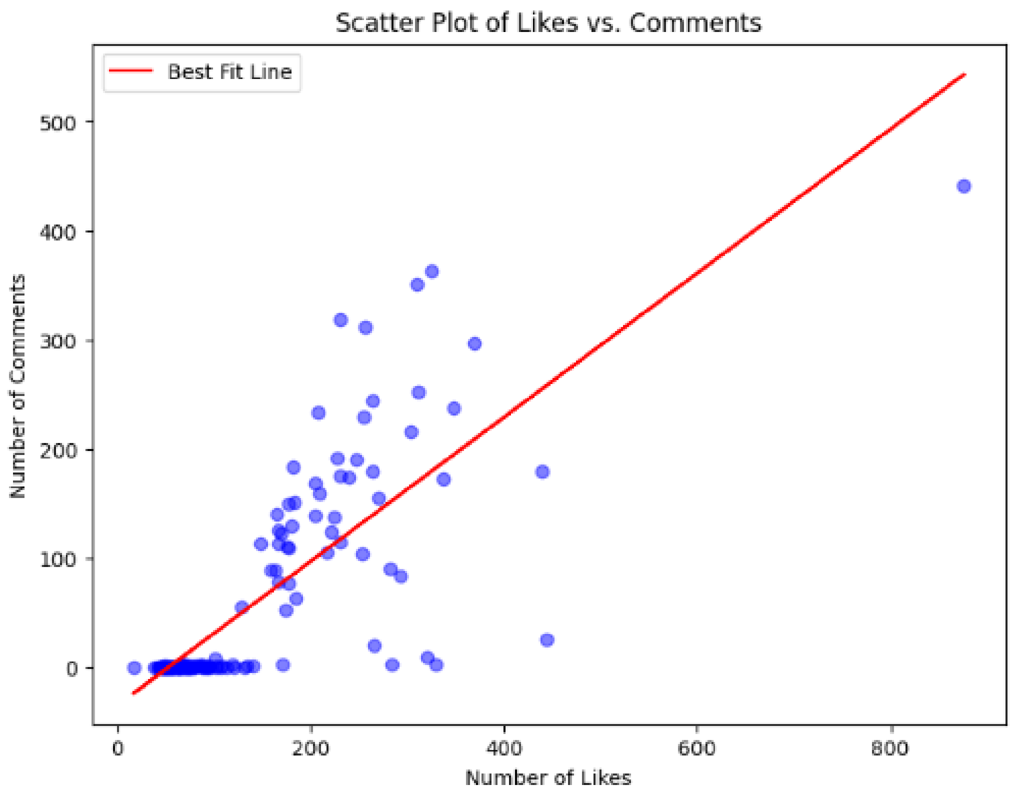
Kernel Density Plot of Like Count and Comment Count

Compare the distributions of two variables, 'Like Count' and 'Comment Count,' in a single visualization. The filled areas under the curves represent the density of data points. Like Comment is generally more count compare to comment.



Scatter Plot of Likes and Comments

Althought there are more likes compared to comments, the positive gradient best fit line show that the likes and comments are positively correlated. The higher the like on the post, the higher the comments.

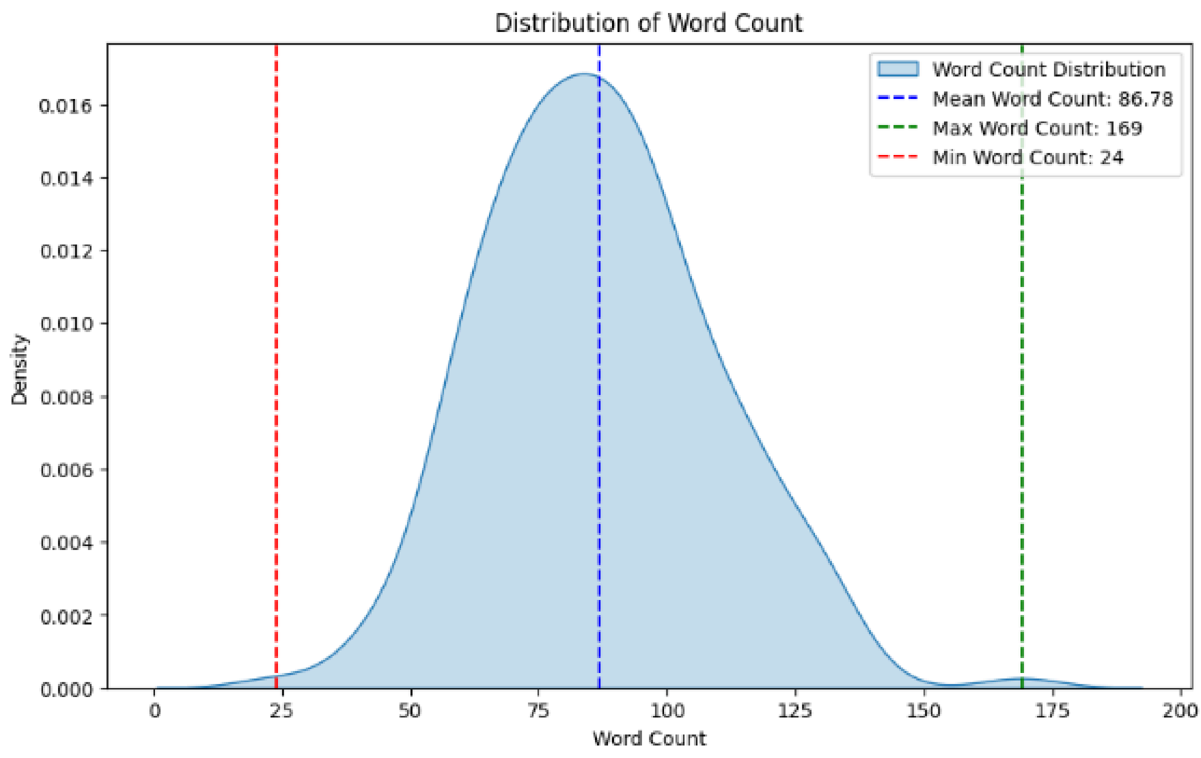


Distribution of Word Count

This plot displays the distribution of word counts of descriptions. The x-axis represents the word count of each description, and the y-axis represents the density of descriptions at each word count.

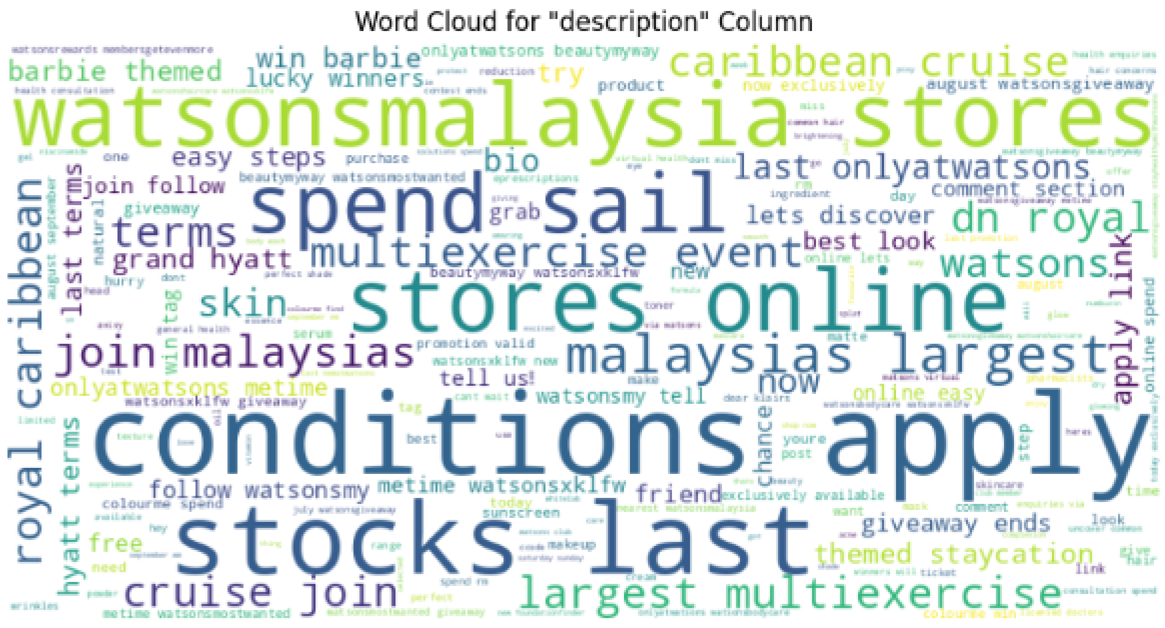
The central peak in the distribution indicates that a significant portion of the descriptions has a word count clustered around a particular value. This central peak represents the most common word count in the dataset, which, in this case, is the mode or typical length of these descriptions.

The blue line represents the mean word count, which is the average length of all descriptions in the dataset.
The green line represents the maximum word count, indicating the longest description present in the dataset.
The red line represents the minimum word count, indicating the shortest description in the dataset.



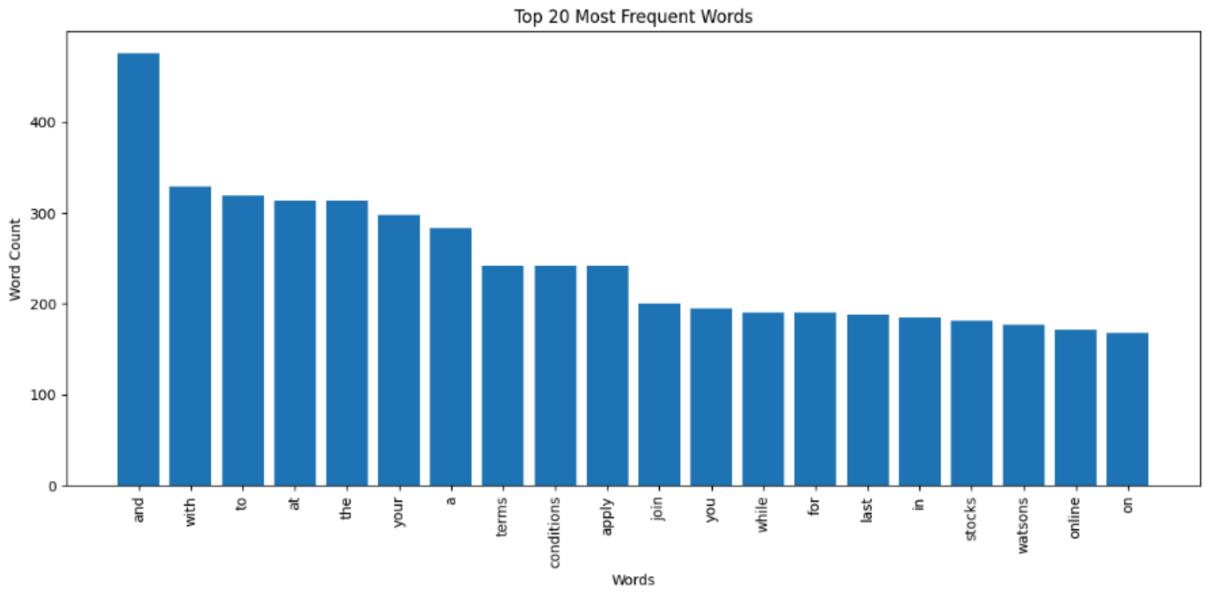
Word Cloud for Description

This word cloud visually represents the most frequently occurring words in the "description" column of your dataset. Words are sized proportionally to their frequency, with larger words indicating higher occurrence in the text. The word cloud provides a quick and intuitive way to identify and visualize the most common themes or keywords present in descriptions, help to gain insights into the prominent topics or subjects within your data.



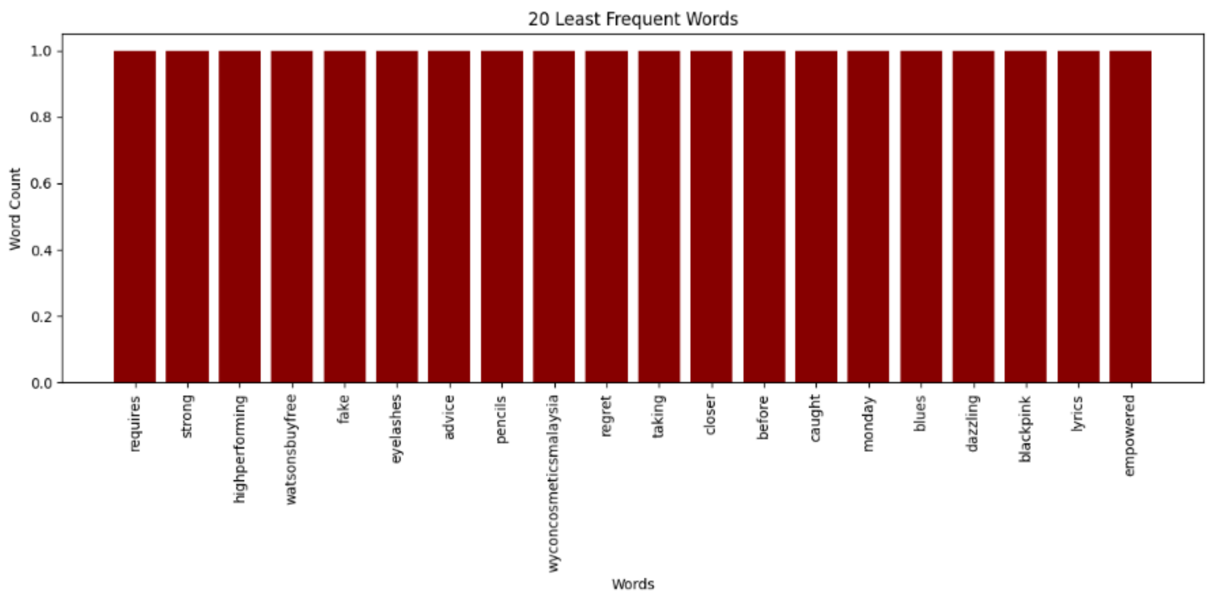
Top 20 Most Frequent Words

This bar plot displays the top 20 most frequently occurring words in the 'description' column of the dataset. The words are sorted in descending order of frequency, with the most common words appearing on the left side of the plot. This visualization provides a quick and clear overview of the most prominent words in the dataset, which can be valuable for understanding the prevalent themes, topics, or keywords within the text data.



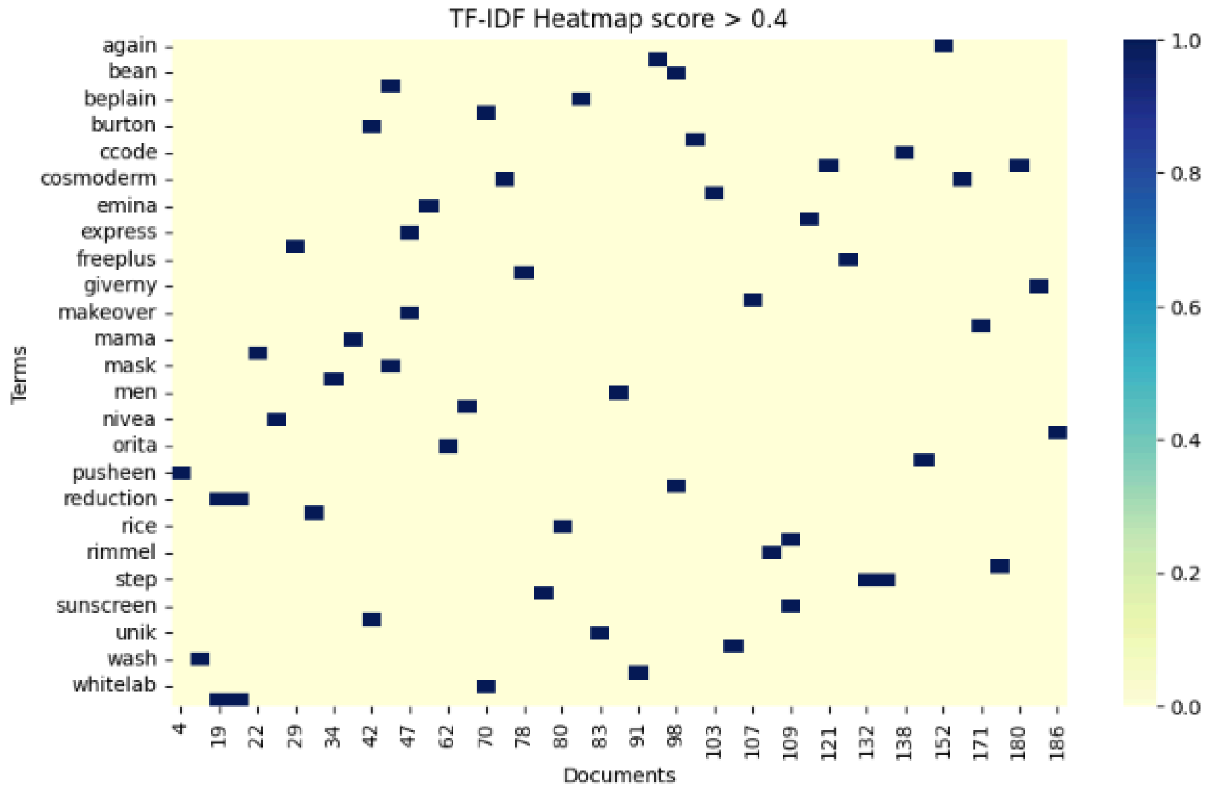
20 Least Frequent Words

This bar plot illustrates the 20 least frequently occurring words in the 'description' column of the dataset. These words are sorted in ascending order of frequency. The visualization highlights words that appear infrequently in the text data, providing insight into less common terms within the dataset.



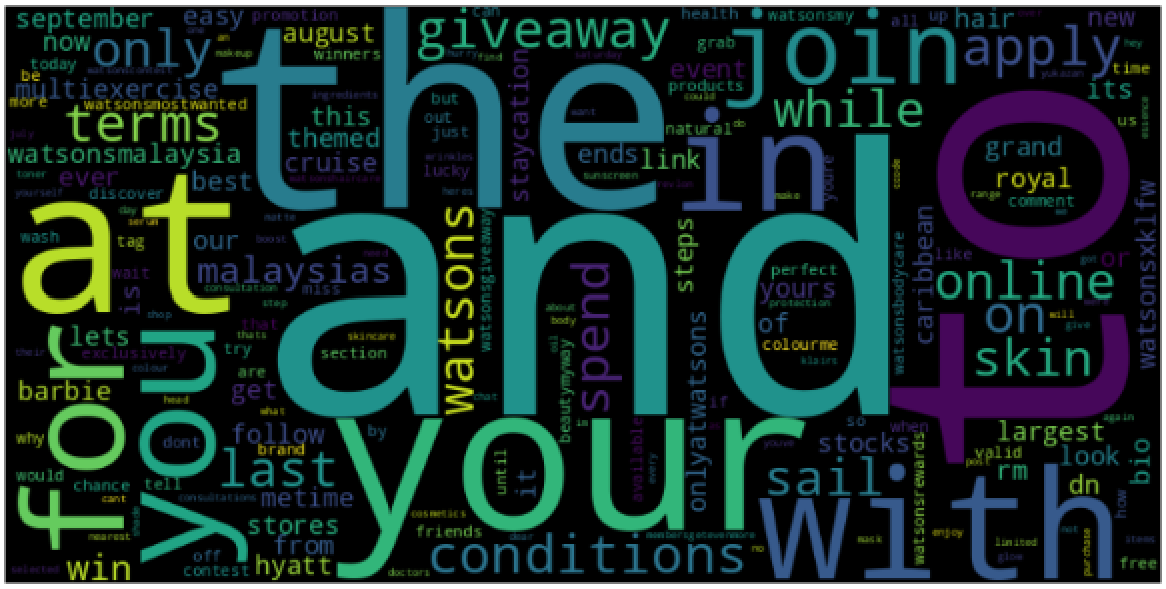
Heatmap of TF-IDF Score greater than 0.4

This heatmap visualizes the TF-IDF (Term Frequency-Inverse Document Frequency) scores for terms in the 'Description' column of the dataset. It focuses on terms with TF-IDF scores greater than 0.4. Each row represents a term, while each column represents a document or text entry. The color intensity in the heatmap indicates the TF-IDF score, with dark blue colors representing higher scores, providing a visual representation of the importance of terms within each document.



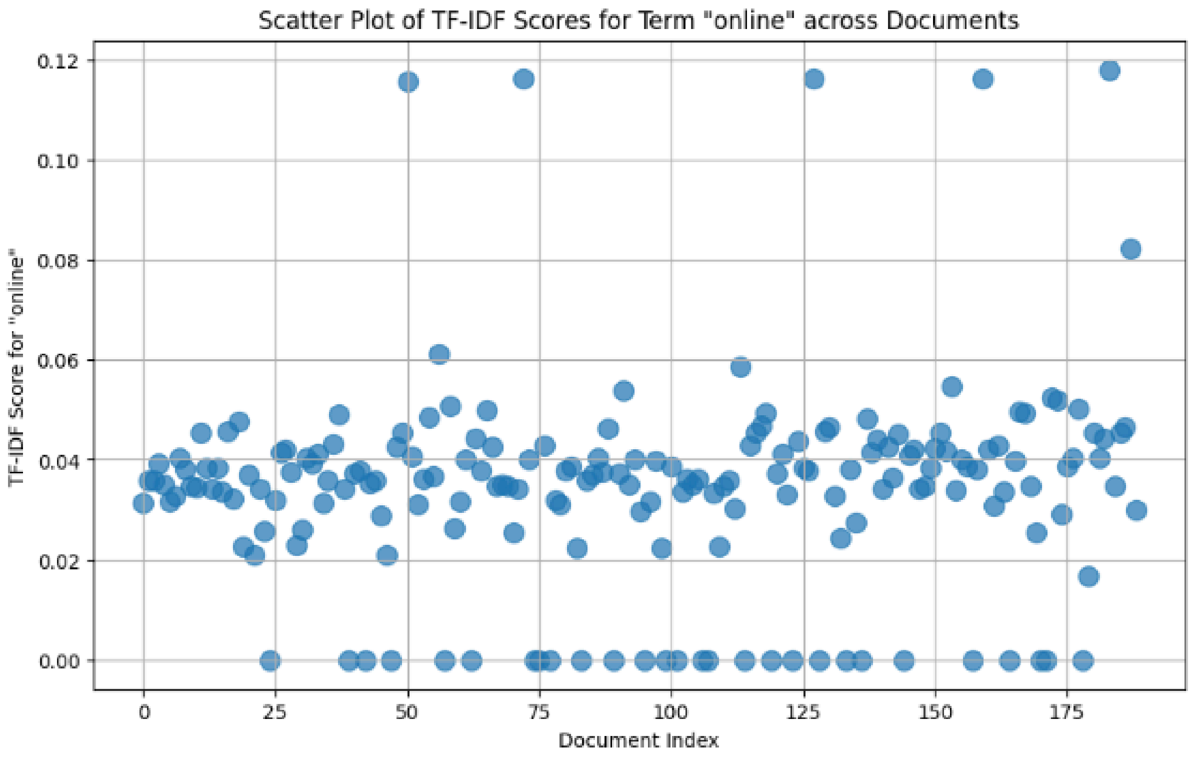
Word cloud of the average TF-IDF scores

This word cloud is generated based on the average TF-IDF (Term Frequency-Inverse Document Frequency) scores for each term across all documents in the dataset. Terms that are more prominent and have higher average TF-IDF scores appear larger in the word cloud, while terms with lower scores appear smaller. It provides a visual summary of the most significant terms within the corpus, emphasizing words that are relatively unique or distinctive across documents and highlighting their importance in representing the overall content of the dataset.



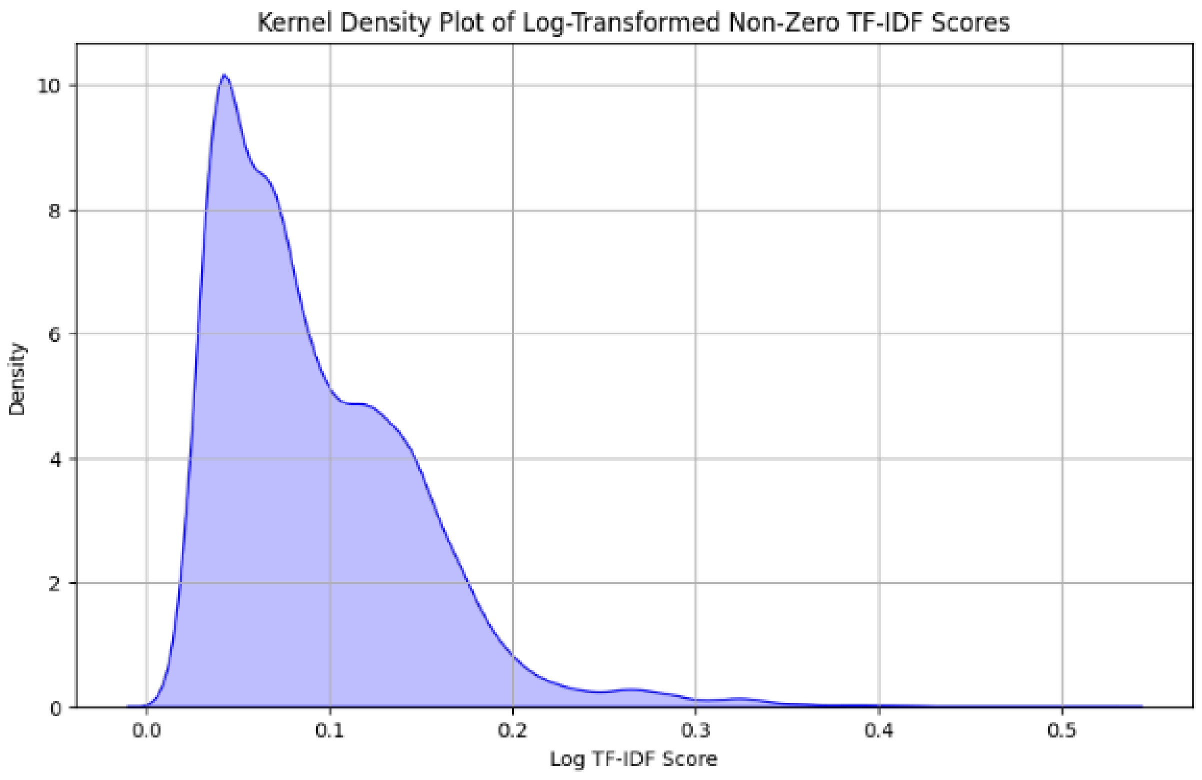
Scatter Plot of TF-IDF Scores for Term “Online” across Document

This scatter plot visualizes the TF-IDF (Term Frequency-Inverse Document Frequency) scores for the term "online" across all documents in the dataset. Each point on the plot represents a document, and its position along the y-axis indicates the TF-IDF score for the term "online" within that document. The scatter plot helps to understand the distribution and variation of the importance of the term "online" across the entire corpus, highlighting which documents place higher or lower emphasis on this term in their content.



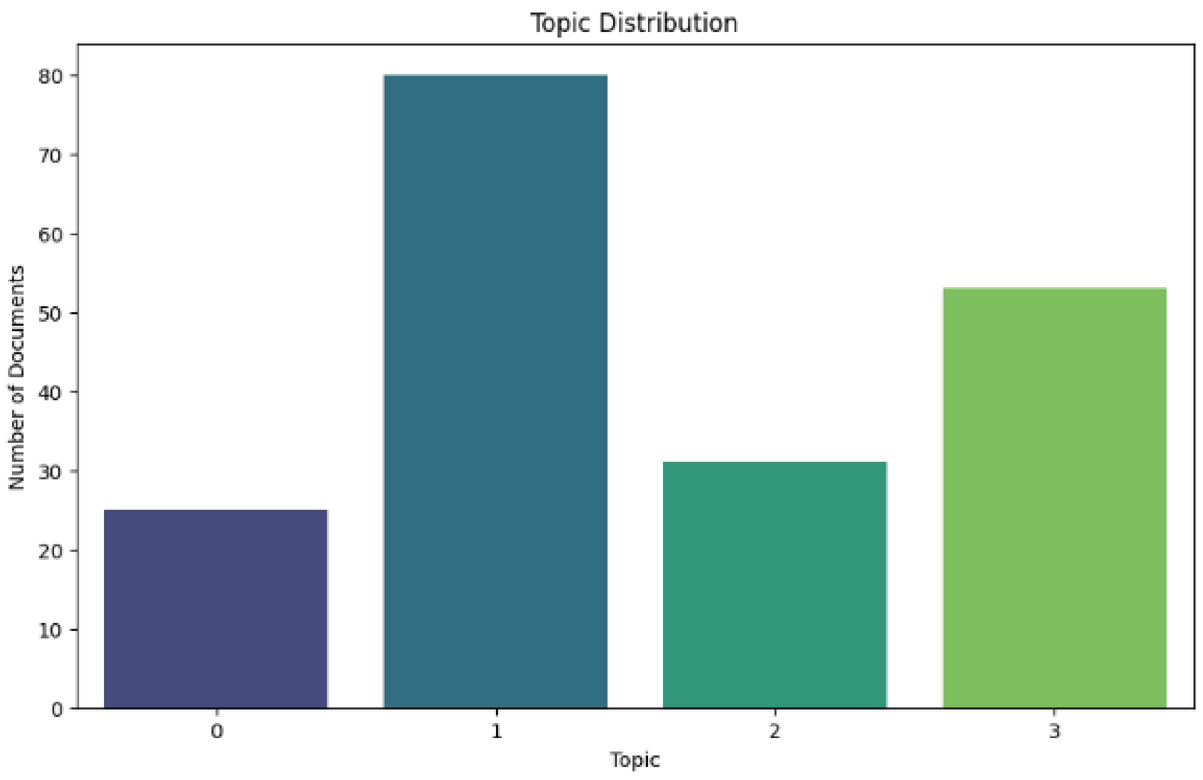
Kernel Density Plot of Log-Transformed Non-Zero TF-IDF Scores

This Kernel Density Plot displays the distribution of log-transformed non-zero TF-IDF (Term Frequency-Inverse Document Frequency) scores from the dataset. The x-axis represents the logarithmic scale of the TF-IDF scores, providing a visual representation of the spread and concentration of importance values for terms within the documents. The plot allows to explore the density of TF-IDF scores after applying the logarithmic transformation, which can help identify patterns and variations in the significance of terms across the text data.



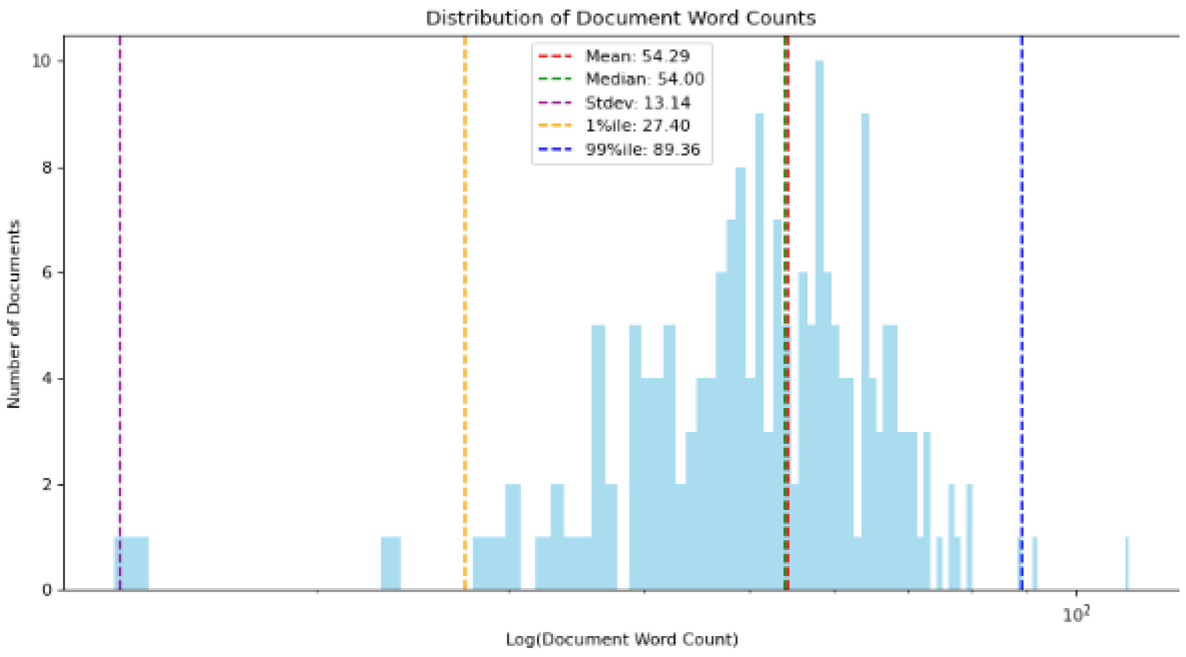
Topic Distribution

This bar plot visualizes the distribution of topics within the dataset, where each topic is represented by a numeric label. The x-axis displays the topic labels, while the y-axis represents the number of documents assigned to each topic. It provides a clear overview of the distribution of documents across different topics, help to understand the prevalence of each topic and the balance or skew in the topic modeling results.



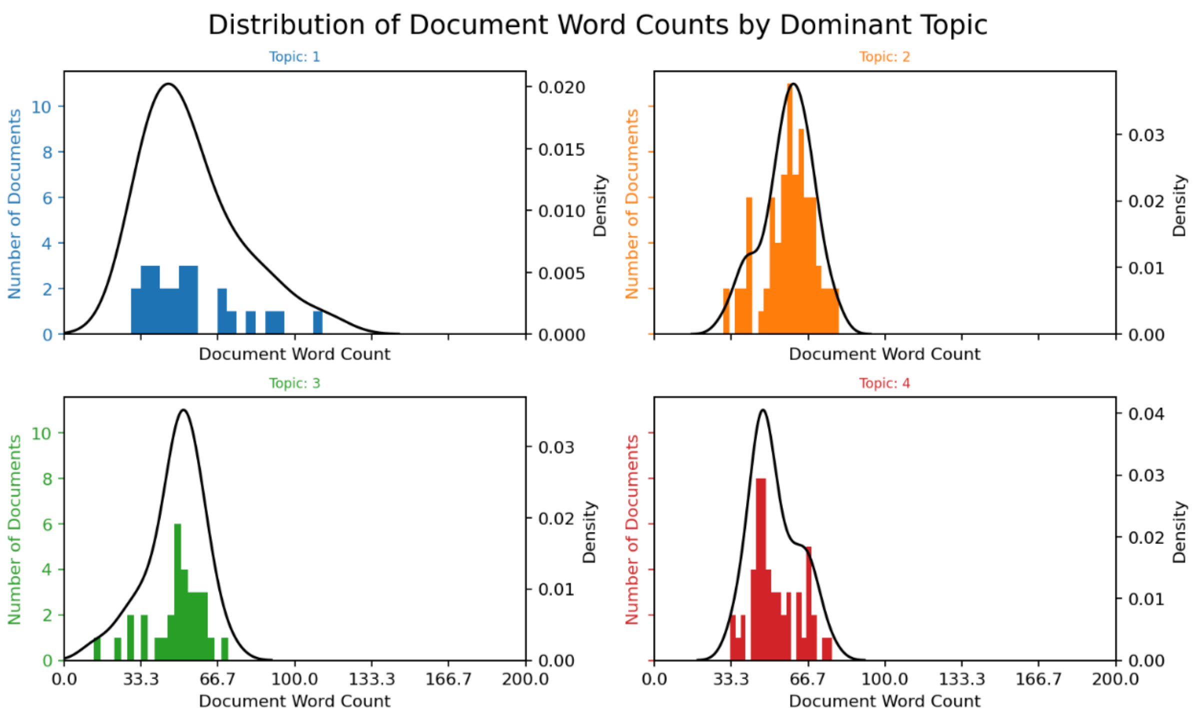
Frequency Distribution of Word Counts in Documents

This histogram plot visualizes the distribution of word counts across a collection of documents, and it uses a logarithmic scale for the x-axis to accommodate a wide range of word counts. Each bar in the histogram represents the number of documents falling within a particular word count range. The plot provides insights into the variability in document lengths, showing that most documents have a relatively low word count, with some outliers having significantly more words. Additionally, the plot includes key statistics such as the mean, median, standard deviation, and selected percentiles to summarize the distribution of document word counts.



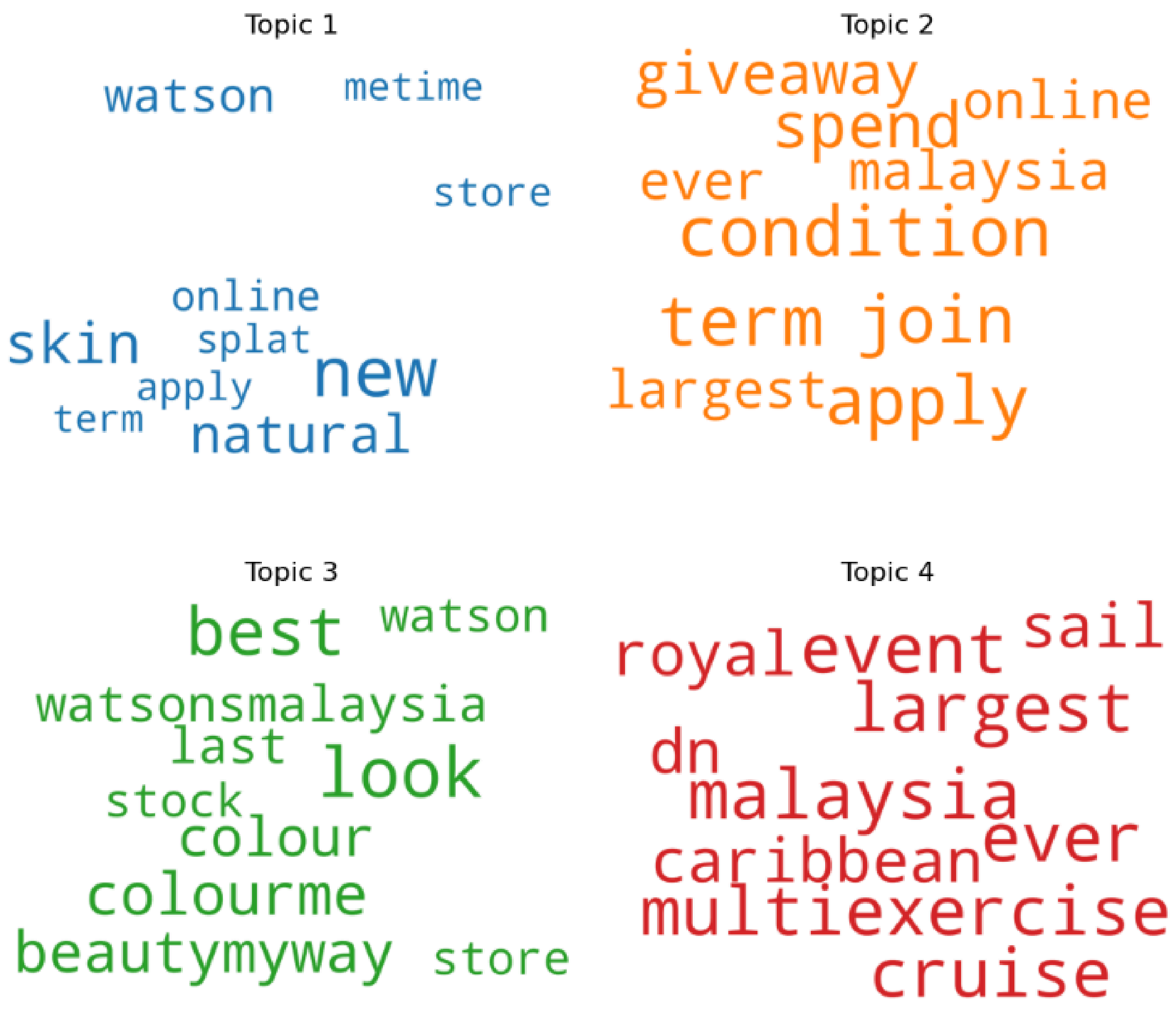
Distribution of Document Word Counts by Dominant Topic

This set of four subplots presents the distribution of document word counts within different topics as part of a topic modeling analysis. Each subplot corresponds to a specific topic, and the histograms show how the word counts are distributed across the documents associated with each topic. The overlaid black KDE (Kernel Density Estimation) curves provide a smooth representation of the same data, highlighting the overall distribution shape. The x-axis limit has been adjusted to 200 words, and the x-axis ticks have been updated to better display the word count distribution for documents within each dominant topic.



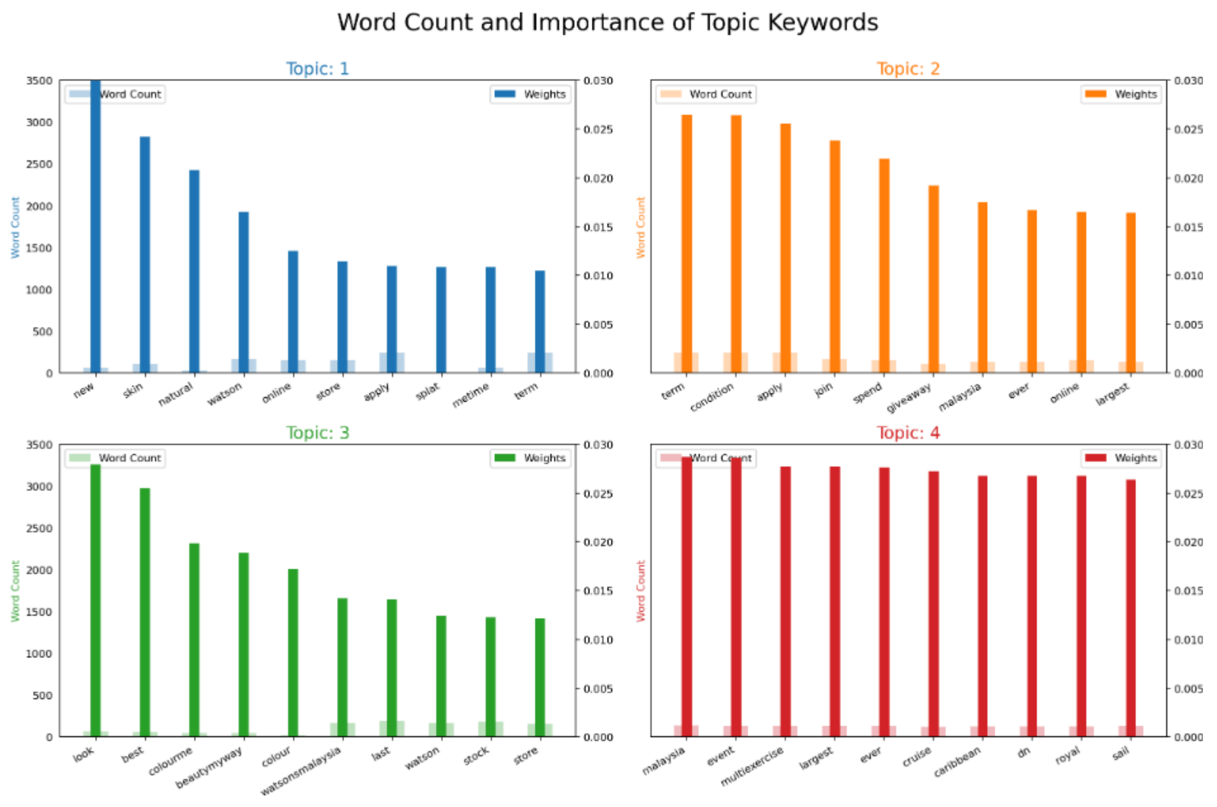
Word Clouds of Top N Keywords in Each Topic

This grid of four subplots displays word clouds representing the top words associated with each of the four topics generated by a Latent Dirichlet Allocation (LDA) model. Each subplot corresponds to one topic and visualizes the most frequent words within that topic. The word clouds use different colors from the 'tab10' colormap to distinguish between topics, and the size of each word in the cloud corresponds to its frequency within the topic, with larger words indicating higher importance. The result provides a visual summary of the key terms that define and differentiate each topic, making it easier to interpret and label them.



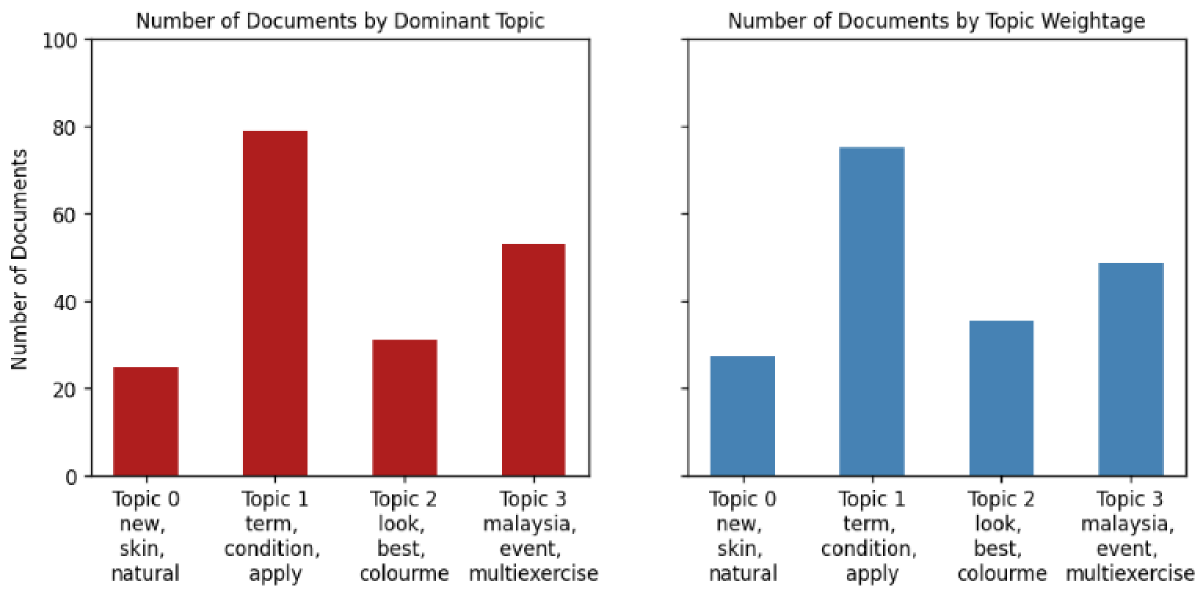
Word Counts of Topic Keywords

This set of four subplots visualizes the word count and importance of topic keywords for the four topics identified by a Latent Dirichlet Allocation (LDA) model. Each subplot corresponds to a different topic, and it provides a comparison of word counts (shown as bar heights) and word importance (shown as bar widths) for the keywords associated with that topic. The bar heights represent the frequency of each word within the topic, while the bar widths indicate the importance of each word within the topic. The visualization allows to understand which words are both frequent and highly important in defining the characteristics of each topic.



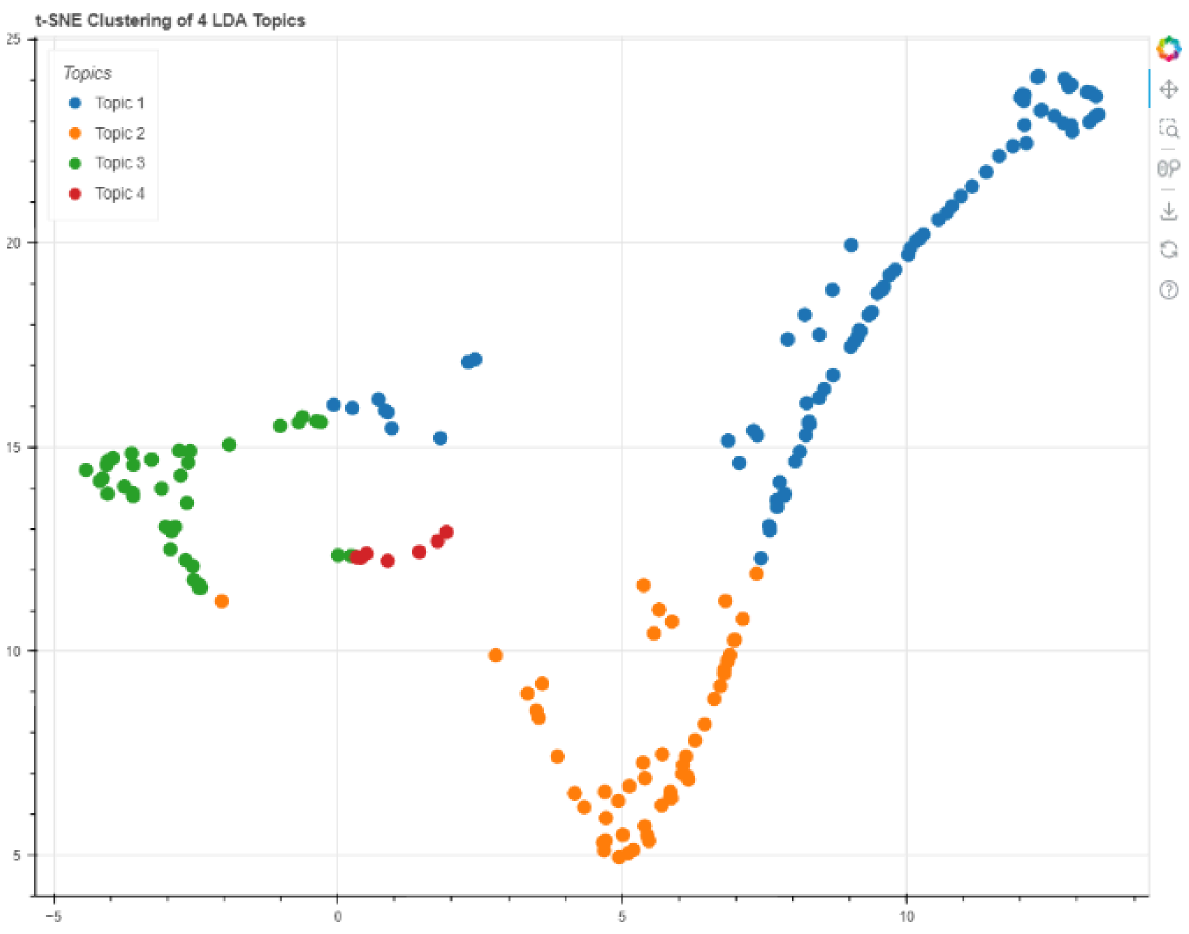
Number of Documents by Dominant Topic and Topic Weightage

This side-by-side bar plot consists of two panels that provide insights into the distribution of topics within a corpus of documents. The first panel on the left shows the number of documents attributed to each dominant topic, with topics labeled on the x-axis and the corresponding document counts on the y-axis. The second panel on the right displays the number of documents distributed by their topic weightage, with topics also labeled on the x-axis and document counts on the y-axis. Both panels offer a visual summary of how documents are distributed across various topics, allowing for a comparison between dominant topics and topic weightage.



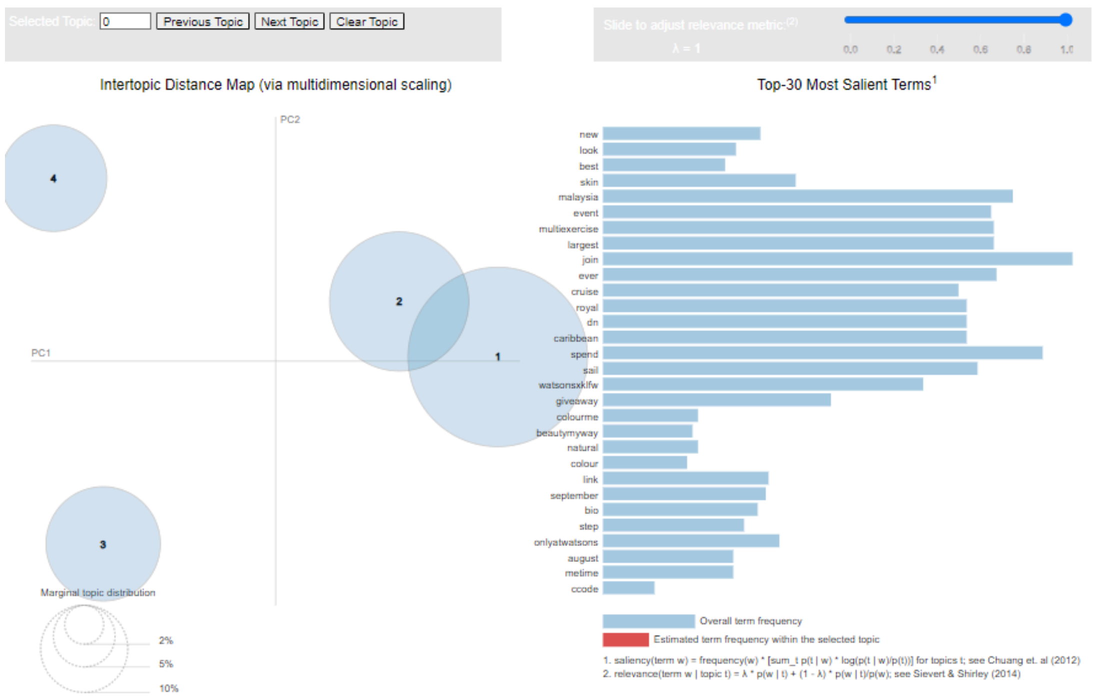
t-SNE Clustering Chart

A t-SNE (t-Distributed Stochastic Neighbor Embedding) clustering chart is a dimensionality reduction technique often used for visualizing high-dimensional data in lower-dimensional space while preserving the relationships between data points. It is a valuable visualization tool for exploring and understanding the results of a topic modeling task. t-SNE can be applied to the topic distribution of documents to reduce the dimensionality of the data and visualize it in a lower-dimensional space while preserving the relationships between documents.



pyLDAVis

PyLDAVis is a Python library used for visualizing and interpreting topic models, particularly those created using Latent Dirichlet Allocation (LDA) or related algorithms. It provides an interactive web-based interface that allows you to explore the topics, their word distributions, and their relationships within a corpus of text data. This tool is valuable for gaining insights into the topics discovered by your topic modeling algorithm and for making the results more accessible to others.



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

References

Apache.org. (2023). *Apache Hive*. [online] Available at: https://hive.apache.org/ [Accessed 16 Sep. 2023].

Fatih Karabiber. (2023). TF-IDF — Term Frequency-Inverse Document Frequency. LearnDataSci. [online] Available at: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/#:~:text=Term%20Frequency%20%2D%20Inverse%20Document%20Frequency%20(TF%2DIDF)%20is,%2C%20relative%20to%20a%20corpus).

GeeksforGeeks. (2019). *Python PoS Tagging and Lemmatization using spaCy*. [online] Available at: https://www.geeksforgeeks.org/python-pos-tagging-and-lemmatization-using-spacy/ [Accessed 16 Sep. 2023].

GeeksforGeeks. (2020). *NLP Expand contractions in Text Processing*. [online] Available at: https://www.geeksforgeeks.org/nlp-expand-contractions-in-text-processing/ [Accessed 16 Sep. 2023].

GeeksforGeeks. (2021). *HBase Model in Hadoop*. [online] Available at: https://www.geeksforgeeks.org/hbase-model-in-hadoop/ [Accessed 16 Sep. 2023].

MonkeyLearn Blog. (2019). Topic Modeling: An Introduction. [online] Available at: https://monkeylearn.com/blog/introduction-to-topic-modeling/ [Accessed 16 Sep. 2023].

Ibm.com. (2023). *What is HBase? | IBM*. [online] Available at: https://www.ibm.com/topics/hbase [Accessed 16 Sep. 2023].

Appendix A Work Distribution Among Members

	Subtasks handled by each team member			
Tasks	Tang Sharren	Koong Jie Lum	Loh Eng Sin	Tham Hiu Huen
Item 6 Extract data	1. Look for a suitable data source and crawl data from Instagram @watsonsmy using PhantomBuster website. 2. Store watsons-dataset.csv in HDFS 3. Setup Kafka producer server to ingest the dataset stored in HDFS and send the messages ingested to a Kafka topic(watsons-topic).	1. Retrieve data from HBase.	1. 2. 3.	1. 2. 3.
Item 7 Condition & transform data	1. In Kafka consumer code, preprocess the text data, including lowercasing, stopwords removal,punctuation removal, and so on. 2. 3.	1. Consumer-watsons.py code to consume messages from a Kafka topic, preprocess the 'description' and ‘postId’ fields 2. Store the processed data from consumer into an HBase table. 3.	1. 2. 3.	1. 2. 3.

Item 8 NLP tool	1. Save the lemmatized & contraction-handled csv into hdfs as 1 file instead of splitting into multiple files using repartition(1) 2. Rename the saved csv generated by repartition(1) in hdfs with a consistent name using code 3. Load the lemmatized & contraction-handled csv into HIVE table	1. Contraction Handler 2. Lemmatizer with POS Tagger 3.	1. 2. 3.	1. 2. 3.
Item 9 Text analytics task	1. 2. 3.	1. 2. 3.	1. 2. 3.	1. Retrieve pre-processed data from hdfs 2. Explorory Data Analysis 3. Perform TF-IDF 4. Perform Topic Modelling 5. Data Visuaization