Graded Challenges 2 (DA3 and DA4)

Name: Helge Moes

Student Number: 11348801

WG: 02

Date: 07-03-2024

Now that we've seen how Python works and and how to use Pandas, it's time for you to combine and apply this knowledge to analyse Twitter data on your selected organisation or topic. In this challenge, you will work with the Twitter data and will execute all the steps necessary for visualising answers to a simple research question.

The challenge consists of three components:

1. **Programming**
2. **Interpretation**
3. **Reflection**

Some important notes for the challenges:

1. While we of course like when you get all the answers right, the important thing is to exercise and apply the knowledge. So we will give some credit for challenges that may not be complete, as long as we see enough effort. The rubric (see Canvas) reflects this.
2. You will deliver the challenges on Canvas. Make sure to follow the turning-in instructions.

Usage of AI-generated content

In line with the course policies, it is **not allowed** to use AI-generated content to any explanations or text provided in MarkDown. This means that for **Interpretation** and **Reflection** students are **not allowed** to use any AI-generated content or AI-assisted tools.

For **Programming**, however, AI-assisted tools (e.g., GitHub Co-Pilot or ChatGPT) may be helpful to understand how to solve some of the questions - as one may also use Google to search for solutions online (e.g., on StackOverflow or other websites). They can, therefore, be used for the programming part of this challenge exceptionally - and only for this part - unless otherwise noted in the question. It is however your responsibility to test and make sure that the solution works - and explain this, in your own words, in the interpretation section.

Facing issues?

We are constantly monitoring the issues on the GitHub to help you out. Don't hesitate to log an issue there, explaining well what the problem is, showing the code you are using, and the error message you may be receiving.

Important: We are only monitoring the repository in weekdays, from 9.30 to 17.00. Issues logged after this time will most likely be answered the next day. This means you should not wait for our response before

submitting a challenge :-)

Programming challenge

You will use an actual Twitter data. To do so, you need to:

- Use the Twitter data available on OneDrive.
- Use sentiment analysis file available on OneDrive

The aim of this challenge is to propose and visualize the answer to a RQ that takes sentiment as the dependent variable and some information about your tweets as independent variable. To be able to visualize the answer, you will need to take a number of steps described below.

Propose a research question that has **sentiment** as dependent variable and some **categorization of your tweets** as independent variable. **Make sure to write down the RQ in your submission**.

The categorization needs to be meaningful for sentiment of tweets. It can be based on any available information in your data. You will need to create the categories in one of the next steps.

1. Load your Twitter dataset and the dataset with sentiment scores of your tweets.
2. Minimize and pseudonymize your Twitter dataset. *Tip: to minimize it, think what variable are necessary for you to answer your RQ.*
3. Merge your tweets dataset with sentiment so that you have a sentiment score for each tweet. *Tip: Check if the length of the dataframe generated by the merge makes sense.*
4. Create one variable that operationalizes the sentiment of tweets (i.e., that somehow aggregates the information of it being positive or negative - or potentially neutral - into one single variable).
5. Write a function that categorizes your tweets. Apply this function to your data.
    - The function can use Natural Language Processing (and categorize them based on the content of the tweets) or other rules (for example, categorize tweets based on engagement, authors etc.).
    - The categorization can be binary (taking the value 0 (when the tweet is not of that category) or 1 (when the tweet is of that category) or categorical with multiple meaningful categories that a tweet can fall under, but needs to contain meaningful information about the tweets.
6. Calculate the reliability of the sentiment analysis **or** of the category created on step 5. To do so, you will need to:
    A. Export a random sample of (at least) 50 tweets to Excel
    B. Manually code this random sample for sentiment **or** for the category created on step 5
    C. Load the manually coded data on Pandas
    D. Merge the manual coding you did with the automated categorization (steps 4 and 5)
    E. Calculate intercoder reliability between your coding, and the automated categorization
7. Visualise and provide descriptive statistics to answer your RQ:
    A. Show descriptive statistics for the IV (categorical variable) and the DV (sentiment)
    B. Create one univariate visualisation for the IV
    C. Create one univariate visualisation for the DV
    D. Create one bivariate visualisation with the IV and DV in the same chart
    E. Show the descriptives of the DV grouped by the IV and interpret them

Using MarkDown

Make sure to combine code and markdown to answer these questions. Mention specifically the question (and question number) and the answer in markdown, relating to the code and the output of the code. Failing to do will impact the grade, as we will not be able to see whether you answered the question.

Wordcount: There is no maximum wordcount for this section.

Research Question

The following research question shall be explored in this sentiment analysis:

RQ: Does the use of CAPSLOCK correlate with the sentiment of the tweet?

With this question, I expect to find a certain sentiment (positive, negative or neutral) that would be behind the use of capslock. For instance, a user might want to highlight certain information or convey they disaproval of a certain post.

In order to answer this research question: the following datasets are used from the OneDrive:

- `google_or_@google.jsonl`
- `google_or_@google_EN_withtext_completed.pkl`

Important: Both of the files need to be in the same folder as the Jupyter Notebook file in order for the cell below to run correctly.

Question 1. Loading libraries

In [95]:
```
# Importing necessary libraries when needed
import pandas as pd
import os
```

In [96]:
```
# To ensure from which directory I am working from
print(os.getcwd())
```

```
/Users/helgegeurtjacobusmoes/Desktop/Digital Analytics
```

The two functions from DA2 - Loading Different Datasets are defined by using them to restructure the Twitter data.

In [97]:
```
def get_public_metrics(row):
    if 'public_metrics' in row.keys():
        if type(row['public_metrics']) == dict:
            for key, value in row['public_metrics'].items():
                row['metric_' + str(key)] = value
    return row

def get_tweets(df):
    if 'data' not in df.columns:
        return None
    results = pd.DataFrame()
    for item in df['data'].values.tolist():
        results = pd.concat([results, pd.DataFrame(item)])

    results = results.apply(get_public_metrics, axis=1)
```

```
    results = results.reset_index()
    del results['index']

    return results
```

In [98]:
```
# Read initial tweets data from a JSON Lines file into a DataFrame
initial_tweets = pd.read_json("google_or_@google.jsonl", lines=True)

# Make sure to use the correct .read_ method depending on the file type
sentiments = pd.read_pickle("google_or_@google_EN_completed.pkl")
```

In [100…
```
# Calling the get_tweets function with the initial_tweets parameter and assigning the re
tweets = get_tweets(initial_tweets)
```

In [131…
```
# Display of tweets
print(f"***\nThere are {len(tweets)} rows of tweets.\n***")
```

```
***
There are 24254 rows of tweets.
***
```

In [107…
```
# Display of the sentiment data
print(f"***\nThere are {len(sentiments)} rows of sentiment scores.\n***")
sentiments.head()
```

```
***
There are 8411 rows of sentiment scores.
***
```

Out[107]:

	id	positive	negative	neutral
4	1622967653991354370	1	-1	0
10	1622967653093588995	2	-2	-1
12	1622967652644974594	1	-1	0
14	1622967651462189057	4	-1	1
15	1622967651437006848	2	-1	1

Question 2. Minimizing and pseudonymizing dataset

Before the data can be merged, it needs to be minimized and pseudonymized.

1. Missing values:

Check if there are values missing in the datasets. This method is retrieved from GitHub week 3 Data Understanding.

In [108…
```
# Print the number of missing values in the tweet dataset
print("Missing values in the tweet dataset:\n", tweets.isna().sum())

# Print the number of missing values in the sentiment dataset
print("\nMissing values in the sentiment dataset:\n", sentiments.isna().sum())
```

```
Missing values in the tweet dataset:
 conversation_id                0
reply_settings                  0
referenced_tweets            3420
edit_history_tweet_ids          0
public_metrics                  0
text                            0
context_annotations         12601
entities                      224
lang                            0
id                              0
edit_controls                   0
author_id                       0
created_at                      0
possibly_sensitive              0
attachments                 22124
in_reply_to_user_id         21399
geo                         24163
withheld                    24253
metric_retweet_count            0
metric_reply_count              0
metric_like_count               0
metric_quote_count              0
metric_impression_count         0
dtype: int64

Missing values in the sentiment dataset:
 id          0
positive    0
negative    0
neutral     0
dtype: int64
```

2. Language tweets:

It appears that not all tweets are in English just by examining the `text` column. Therefore, I will examine this by using the `.value_counts()` method on the `lang` column. This method was retrieved from Data Understanding.

In [109…

```
# Counting each unique value in the "lang" column of the DataFrame "tweets"
tweet_lang_counts = tweets["lang"].value_counts()
print(tweet_lang_counts)
```

```
lang
en     8411
tr     5548
ja     3898
es     1059
in      906
th      871
ko      829
ar      403
pt      379
qme     344
fr      337
zxx     290
und     228
hi      100
zh       98
de       75
sv       66
it       61
ca       46
tl       46
pl       31
nl       31
vi       23
ro       21
et       20
ta       17
ru       16
cs       14
da       13
ht       12
fa       11
el        7
iw        7
qam       5
cy        4
uk        3
qht       3
fi        3
ur        3
lv        2
lt        2
bg        2
no        2
eu        2
te        2
kn        1
bn        1
ne        1
Name: count, dtype: int64
```

For this challenge, I shall mainly focus on English tweets. Therefore, I shall select these specific tweets by slicing, which results in a data frame that consists of only english tweets. This idea correlates with what was executed in 12_0_DataPreparationVisualisation.

In [110…

```
# Select only English Tweets
tweets_eng = tweets[tweets["lang"]=="en"]
print(f"The initial dataset contains {len(tweets)} rows of tweets.")
print(f"The english dataset contains {len(tweets_eng)} rows of tweets.")
print(f"\nCheck if all the 8411 tweets are indeed in english:\n\n {tweets_eng['lang'].va
```

```
The initial dataset contains 24254 rows of tweets.
The english dataset contains 8411 rows of tweets.

Check if all the 8411 tweets are indeed in english:

 lang
en    8411
Name: count, dtype: int64
```

After examining the overlap in `id` in question 1, it appears that all English tweets in this dataset possess a sentiment score. Thus, this step is considered as completed.

3. Selecting relevant columns:

In order to answer the research question, not all columns are relevant. Therefore, I can minimize the dataframe by selecting columns by slicing. This was retrieved from 11_DataPrivacy.

The necessary variables to address my research question are the `text` and `id`, which are essential for merging with the sentiment dataset.

In [111…
```
# Access the column names of the DataFrame "tweets_eng"
tweets_eng_columns
```

Out[111]:
```
Index(['conversation_id', 'reply_settings', 'referenced_tweets',
       'edit_history_tweet_ids', 'public_metrics', 'text',
       'context_annotations', 'entities', 'lang', 'id', 'edit_controls',
       'author_id', 'created_at', 'possibly_sensitive', 'attachments',
       'in_reply_to_user_id', 'geo', 'withheld', 'metric_retweet_count',
       'metric_reply_count', 'metric_like_count', 'metric_quote_count',
       'metric_impression_count'],
      dtype='object')
```

In [112…
```
# Selecting interesting columns
tweets_rq = tweets_eng[["id", "text"]]

tweets_rq.head()
```

Out[112]:

	id	text
4	1622967653991354370	RT @enhypenupdates: [EN-LIVE RECAP 🔊]\n\n230203 “In MANILA”\n🔊: https://t.co/I5TxpcOlAf\n\n230204 “Tired and Sleepy Mildangz”\n🔊: https://t.co/vZ…
10	1622967653093588995	RT @BlurCommunity_: T-minus 7🔥 days! Are you ready BLUR Community?\n\nblur list $BLUR on pancakeswap\n\n#OKX #Huobi #Bybit #Gateio #Bitget #Bi…
12	1622967652644974594	@midtrophic you can always google tutorials or feel free to ask me
14	1622967651462189057	RT @martian_wallet: Wizards here??? Amazing giveaway with @WizardLandSui🧙 \n\nWe are giving away 10 WL and 10 OG spots!🔥 \n\n1️⃣Follow both Twi…
15	1622967651437006848	RT @TENzenter: Project for TENLEE \nTEN THAILAND x TENzenter\n\n7 YEARS 10GETHER\n\nDONATION SUPPORT \n🔽 https://t.co/ZexfwX3aMp\n\nLET'S LIGHTE…

4. Anonimizing the data:

Since the user names are still present in the `text` column, it needs to be pseudonimized in order to avoid privacy issues. Consequently, regular expressions are used in order to manipulate the 'text' column with the `.replace()` method retrieved from 11_DataPrivacy.

This is executed by replacing all Twitter mentions (strings starting with '@') with '@user' to anonymize them. It first replaces all patterns matching '@\S+' (where '\S+' matches one or more non-whitespace characters @user', and then removes all occurrences of '@user'.

In [113…

```
# Importing the regex library
import regex as re

# Anonimizing and removing the @mention tag.
tweets_rq_anon = tweets_rq.replace(
    '@\S+', '@user', regex = True).replace(
    '@user', '', regex=True)

# Checking whether anonmiziation worked:
tweets_rq_anon["text"].sample(5)
```

Out[113]:

```
3303
RT  [□□MY GO] NA又怎样 Cupid NA □ 10cm jaemin doll by \n\n□RM50\n□~9 Feb, 6pm  \n\n➡ htt
ps://t.co/Nl1DTtt87B\n\n#dividedforGO ht…
12652
RT  Just noticed booking leads are now available for legal verticals on Google Local Se
rvice Ads (LSAs) previously they were on…
15345
RT  Calling all content creators! \n\nJoin the Web3 creator economy and be a part of so
mething exciting. \n\nFill out our application…
7713
RT  [ANNOUNCEMENT]\n\nGood day, ENGENEs! \n\nWe're about to celebrate the birthday of o
ur leader. We're seeking for committed and e…
4378     To recap: Microsoft are most likely going to announce their new integration wi
th ChatGPT tonight, the day before Google announces Bard..□ Here are some thoughts and
predictions for this year in search and AI: 1/2
Name: text, dtype: object
```

5. Pseudonimizing the data:

In order to pseudonimize the data, all original `id` need to be removed. By using `.reset_index()` and `rename()`, this is realized by assigning a new `id` to each tweet, which makes them unidentifiable. This is retrieved from 11_DataPrivacy.

In [114…

```
# Creating a new unique id for each tweet.
tweets_rq_anon = tweets_rq_anon.reset_index()

# Renaming the 'index' column to 'pseudID' in the DataFrame 'tweets_rq_anon'
tweets_rq_anon = tweets_rq_anon.rename(columns={'index': 'pseudID'})

# Displaying the first few rows of the DataFrame 'tweets_rq_anon' after renaming the col
tweets_rq_anon.head()
```

Out[114]:

	pseudID	id	text
0	4	1622967653991354370	RT [EN-LIVE RECAP □]\n\n230203 “In MANILA”\n□: https://t.co/I5TxpcOIAf\n\n230204 “Tired and Sleepy Mildangz”\n□: https://t.co/vZ…
1	10	1622967653093588995	RT T-minus 7□ days! Are you ready BLUR Community?\n\nblur list $BLUR on pancakeswap\n\n#OKX #Huobi #Bybit #Gateio #Bitget #Bi…
2	12	1622967652644974594	you can always google tutorials or feel free to ask me
3	14	1622967651462189057	RT Wizards here??? Amazing giveaway with \n\nWe are giving away 10 WL and 10 OG spots!□ \n\n1□Follow both Twi…
4	15	1622967651437006848	RT Project for TENLEE \nTEN THAILAND x TENzenter\n\n7 YEARS 10GETHER\n\nDONATION SUPPORT \n□□ https://t.co/ZexfwX3aMp\n\nLET'S LIGHTE…

Question 3. Merging dataframes

In order to merge the data types and the lengths of the tweets and the sentiment scores have to be aligned (10_DataUnterstanding). When the data matches, a merge is performed with `inner`. This allows for an intersection of keys from both frames, which means that that there should be 8411 tweets in the `merged_df`. This method was retrieved from Python Documentation.

In [115…
```
# Printing the length of the tweet data frame
print("Length of the tweet data frame:", len(tweets_rq_anon))

# Printing the length of the sentiment data frame
print("Length of the sentiment data frame:", len(sentiments))

# Printing the data types of columns in the tweets_rq_anon DataFrame
print(tweets_rq_anon.dtypes, "\n")

# Printing the data types of columns in the sentiments DataFrame
print(sentiments.dtypes)
```

```
Length of the tweet data frame: 8411
Length of the sentiment data frame: 8411
pseudID     int64
id         object
text       object
dtype: object

id          object
positive    object
negative    object
neutral     object
dtype: object
```

In [116…
```
# Merging the two dataframes into a third, merged data frame.
merged_df = tweets_rq_anon.merge(sentiments, on='id')

# Checking the length of the merged data frame.
len(merged_df)
```

Out[116]:
```
8411
```

In [117…
```
# Calculating the sum of missing values (NaNs) in each column of the DataFrame merged_df
merged_df.isna().sum()
```

Out[117]:
```
pseudID     0
id          0
text        0
positive    0
negative    0
neutral     0
dtype: int64
```

Question 4. Creating a variable that operationalizes the sentiment of tweets

Before a fourth variable that summarizes the sentiment can be created, I examine whether the distribution and values of the `positive`, `negative` and `neutral` columns with the `describe()` method retrieved from 10_DataUnderstanding).

In Question 3, the three columns seem to have the `object` type, which means that they are read by Python as a string instead of a number. Therefore, the types need to be changed by using `o_numeric)`. This method was retrieved from DA3 Solutions Weekly Challenges).

In [118…
```
# Correcting the data type of the sentiment scores into numerical values
final_df = merged_df[["positive", "negative", "neutral"]].apply(pd.to_numeric)
final_df.dtypes
```

Out[118]:
```
positive    int64
negative    int64
neutral     int64
dtype: object
```

In [119…
```
# Selecting columns for conversion to numeric values, excluding the "text" and "id" colu
numeric_columns = ["pseudID", "positive", "negative", "neutral"]
final_df[numeric_columns] = merged_df[numeric_columns].apply(pd.to_numeric)

# Converting "pseudID" and "text" columns to object data type
final_df[["pseudID", "text"]] = merged_df[["pseudID", "text"]].astype(object)

# Displaying data types of columns in the final DataFrame
final_df.dtypes
```

Out[119]:
```
positive     int64
negative     int64
neutral      int64
pseudID     object
text        object
dtype: object
```

In [75]:
```
# Checking the final_df DataFrame
final_df
```

Out[75]:

	positive	negative	neutral	pseudID	text
0	1	-1	0	0	RT [EN-LIVE RECAP □]\n\n230203 “In MANILA”\n□...
1	2	-2	-1	1	RT T-minus 7□ days! Are you ready BLUR Commu...
2	1	-1	0	2	you can always google tutorials or feel free ...
3	4	-1	1	3	RT Wizards here??? Amazing giveaway with \n\...
4	2	-1	1	4	RT Project for TENLEE \nTEN THAILAND x TENzen...
...	...	...	...	...	...
8406	1	-1	0	8406	RT Partnership with Tocen Launchpad\n✨1 OAT =...
8407	2	-1	1	8407	Nice project\n
8408	1	-1	0	8408	A study on the effects of using organic vs non...
8409	1	-1	0	8409	RT Partnership with Tocen Launchpad\n✨1 OAT =...
8410	2	-1	1	8410	RT Project for TENLEE \nTEN THAILAND x TENzen...

8411 rows × 5 columns

In [120…
```
# Displaying the descriptive statistics
final_df[["negative", "positive", "neutral"]].describe()
```

Out[120]:

	negative	positive	neutral
count	8411.000000	8411.000000	8411.000000
mean	-1.275116	1.548805	0.138985
std	0.605306	0.743479	0.720065
min	-5.000000	1.000000	-1.000000
25%	-1.000000	1.000000	0.000000
50%	-1.000000	1.000000	0.000000
75%	-1.000000	2.000000	1.000000
max	-1.000000	5.000000	1.000000

Based on the results, the variables `positive` and `negative` have a similar range. The summary of the sentiment scores are created with `sum_senti`, which ranges from -5 (very negative) to 5 (very positive). In this case, 0 is considered as neutral. This is retrieved from Laminar Insight.

In [121…

```
# Calculating the sum of sentiment scores for each row and storing the result in a new c
final_df['sum_senti'] =  final_df['positive'] + final_df['negative'] + final_df['neutral

# Displaying descriptive statistics for the 'sum_senti' column
final_df['sum_senti'].describe()
```

Out[121]:

```
count    8411.000000
mean        0.412674
std         1.638298
min        -5.000000
25%         0.000000
50%         0.000000
75%         2.000000
max         5.000000
Name: sum_senti, dtype: float64
```

In [122…

```
# Displaying the first few rows of the DataFrame 'final_df'
final_df.head()
```

Out[122]:

	positive	negative	neutral	pseudID	text	sum_senti
0	1	-1	0	4	RT [EN-LIVE RECAP]\n\n230203 “In MANILA”\n: https://t.co/I5TxpcOlAf\n\n230204 “Tired and Sleepy Mildangz”\n: https://t.co/vZ…	0
1	2	-2	-1	10	RT T-minus 7 days! Are you ready BLUR Community?\n\nblur list $BLUR on pancakeswap\n\n#OKX #Huobi #Bybit #Gateio #Bitget #Bi…	-1
2	1	-1	0	12	you can always google tutorials or feel free to ask me	0
3	4	-1	1	14	RT Wizards here??? Amazing giveaway with \n\nWe are giving away 10 WL and 10 OG spots! \n\n1Follow both Twi…	4
4	2	-1	1	15	RT Project for TENLEE \nTEN THAILAND x TENzenter\n\n7 YEARS 10GETHER\n\nDONATION SUPPORT \n https://t.co/ZexfwX3aMp\n\nLET'S LIGHTE…	2

Question 5. Writing a function that categorizes tweets

To answer the research question, the tweets were categorized based on their use of capslock and whether it was significantly related to the accompanied sentiment score.

Firstly, I want to examine the content of the tweets by using `set_option('max_colwidth', None)` (Python Pandas - Options and Customization, n.d.) and `.sample()` ("Python | Random.Sample() Function," 2018).

In [123…
```
# Setting the option to display the full content of the 'text' column
pd.set_option('max_colwidth', None)

# Sampling 5 random rows from the 'text' column of the DataFrame 'final_df' and displayi
final_df["text"].sample(5)
```

Out[123]:
```
3577
RT  In the last few weeks:\n- https://t.co/FsWd3m8sYg has replaced Google for me.\n- Po
e has replaced ChatGPT.\n- Promptable has…
801
RT  giveaways □□\n#JAEHYUN fanart wallpapers (PC)\n\n□□□□□ — ily\n\n□ :  https://t.co/K
KLqvWPTPp\n\ncommercial not allowed\ndon’t edi…
2879    CBSE Admission 2023 for Classes 9th, 10th, 11th, and class in Dubai. Apply now
#CBSE_Admission #CBSE_Admission_9th_Class #CBSE_Admission_10th_Class #CBSE_Admission_11
th_Class #CBSE_Admission_12th_Class\nhttps://t.co/TcLDZlI94Q\nhttps://t.co/Z3W3P4WjQw\n
https://t.co/l26Ry9pbKP
280
RT  Current Top Cryptocurrency Google Search Results:\n#1 #Bitcoin   : 563M\n#2 #PiNetw
ork : 482M\n#3 #Ethereum : 144M\n#4 #Poly…
6594
RT  Hi I have just published a new blog about an "Unpatched flaw in Google Meet that al
lows unauthorized individuals to join…
Name: text, dtype: object
```

At first glance, I want to remove `\n` by using the `.sub()` method, which gets rid of parts of the string with a regular expression (Arcila, 2022). Moreover, the links are also removed from the tweets in a similar fashion.

Additionally, I will be removing the `RT` tag from the beginning of each retweeted tweet. Although it's in capslock, it doesn't contribute meaningful information to the tweet's content concerning the research question.

When the cleaned text is created, the `positive`, `negative` and `neutral` colunmns can also be removed from `final_df`.

In [124…
```
# Applying regex to remove "RT" tag, newline characters "\n", and URLs from tweets
final_df["text_clean"] = final_df["text"].apply(lambda x: re.sub(r"^RT\s+|\n|https?://\S

# Dropping the sentiment columns "positive", "negative", and "neutral"
final_df.drop(columns=["positive", "negative", "neutral"], inplace=True)

# Displaying a random sample of 5 tweets from the "text_clean" column to verify the resu
final_df["text_clean"].sample(5)
```

Out[124]:
```
4423                                  Check out my Gig on Fiverr: I will do organic youtub
e video promotion by google ads method  #櫻坂46Cool #夕暮れに手をつなぐ  #大奥リアタイ #อิงล็อ
ตxCEOแป้งวิงค์ไวท์ #ククルカン #ワイのLOVEゲーム #虹色開花ポートレート #スリザリン #すり抜け #メンシ開
設
6637      Awesome, thanks! mine is Forerunner 245 which I think is compatible with mos
t, tho might be more than one interface, not sure. Will have a look down Tottenham Cour
t Road tomorrow but can’t find in-stock anywhere via Google.
3153
Google #Bard va-t-il “clubhousiser” #ChatGPT ? #meerkateffect
5537
 CLAIM OAT to get $TOCE from TOCEN &amp;amp; SUIWORLD 1 OAT = 50 $TOCE token airdrop (dis
tributed on mainnet) Claim free OAT: h…
8022
K man spa, big male spa, bts Saphan Khwai, exit 2  There are more than 50 male employee
s to choose from.  Shop line and res…
Name: text_clean, dtype: object
```

In [127…
```
# Displaying the first five rows of the DataFrame final_df for inspection after cleaning
final_df.head()
```

Out[127]:

	pseudID	text	sum_senti	text_clean
0	4	RT [EN-LIVE RECAP]\n\n230203 “In MANILA”\n: https://t.co/I5TxpcOlAf\n\n230204 “Tired and Sleepy Mildangz”\n: https://t.co/vZ…	0	[EN-LIVE RECAP]230203 “In MANILA”: 230204 “Tired and Sleepy Mildangz”:
1	10	RT T-minus 7 days! Are you ready BLUR Community?\n\nblur list $BLUR on pancakeswap\n\n#OKX #Huobi #Bybit #Gateio #Bitget #Bi…	-1	T-minus 7 days! Are you ready BLUR Community?blur list $BLUR on pancakeswap#OKX #Huobi #Bybit #Gateio #Bitget #Bi…
2	12	you can always google tutorials or feel free to ask me	0	you can always google tutorials or feel free to ask me
3	14	RT Wizards here??? Amazing giveaway with \n\nWe are giving away 10 WL and 10 OG spots! \n\n1Follow both Twi…	4	Wizards here??? Amazing giveaway with We are giving away 10 WL and 10 OG spots! 1Follow both Twi…
4	15	RT Project for TENLEE \nTEN THAILAND x TENzenter\n\n7 YEARS 10GETHER\n\nDONATION SUPPORT \n https://t.co/ZexfwX3aMp\n\nLET'S LIGHTE…	2	Project for TENLEE TEN THAILAND x TENzenter7 YEARS 10GETHERDONATION SUPPORT LET'S LIGHTE…

Since the `text_clean` column seems to be cleaned properly, the `text` column has become irrelevant to include in the dataframe and therefore it shall also be removed.

In [129…
```
# Selecting specific columns ("pseudID", "text_clean", "sum_senti") from the DataFrame f
final_df = final_df[["pseudID", "text_clean", "sum_senti"]]
final_df.head()
```

Out[129]:

	pseudID	text_clean	sum_senti
0	4	[EN-LIVE RECAP]230203 “In MANILA”: 230204 “Tired and Sleepy Mildangz”:	0
1	10	T-minus 7 days! Are you ready BLUR Community?blur list $BLUR on pancakeswap#OKX #Huobi #Bybit #Gateio #Bitget #Bi…	-1
2	12	you can always google tutorials or feel free to ask me	0
3	14	Wizards here??? Amazing giveaway with We are giving away 10 WL and 10 OG spots! 1Follow both Twi…	4
4	15	Project for TENLEE TEN THAILAND x TENzenter7 YEARS 10GETHERDONATION SUPPORT LET'S LIGHTE…	2

Function that categorizes the use of capslock in text: caps_cat

This code defines a function named `caps_cat` that categorizes the use of capslock in a given text input.

The function categorizes the capslock use in a text into three categories: low, medium, and high, based on predefined thresholds as a percentage of the total character count.

Here's a breakdown of what each part of the code does:

1. **Counting the total number of characters:**
   - `count_char = len(text)` calculates the total number of characters in the input text and assigns it to the variable `count_char`.

1. **Counting the total number of characters that are in capslock:**
   - `caps_count = sum(1 for char in text if char.isupper())` counts the number of characters in the input text that are in uppercase (capslock) and assigns it to the variable `caps_count`.
   - It uses a generator expression along with the `sum()` function to count the characters in capslock. The generator expression iterates over each character in the text and checks if it is uppercase using the `.isupper()` method.

1. **Defining thresholds:**
   - `high_threshold = count_char * 0.5` defines a threshold value for high capslock use as 50% of the total character count.
   - `low_threshold = count_char * 0.1` defines a threshold value for low capslock use as 10% of the total character count.

1. **Categorizing the caps_count:**
   - The function categorizes the `caps_count` based on the defined thresholds:
     - If the `caps_count` is less than the `low_threshold`, it returns 'low'.
     - If the `caps_count` is between the `low_threshold` and `high_threshold`, it returns 'medium'.
     - If the `caps_count` is equal to or greater than the `high_threshold`, it returns 'high'.

In [193…

```
def caps_cat(text):
    #Counting the total number of characters
    count_char = len(text)

    # Counting the total number of characters that are in capslock
    caps_count = sum(1 for char in text if char.isupper())

    # Defining threshholds
    high_threshold = count_char * 0.5
    low_threshold = count_char * 0.1

    # Categorizing the caps_count
    if caps_count < low_threshold:
        return 'low'
    elif low_threshold <= caps_count < high_threshold:
        return 'medium'
    else:
        return 'high'
```

Since the function has been created, I use the `.apply()` method on the `text_clean` column as retrieved from Week 3 - Data Understanding.

In [136…
```
# Copying the data frame
categorized_df = final_df.copy()

# Applying the new function to the whole column of the data frame
categorized_df['caps_cat'] = final_df['text_clean'].apply(caps_cat)
```

Finally, the categorization is checked whether it has worked for each value of `low`, `medium` and `high`.

In [189…
```
# Filtering rows in the DataFrame where the 'caps_cat' column has the value 'low'
low_capslock_df = categorized_df[categorized_df['caps_cat'] == 'low']

# Displaying the first few rows of the filtered DataFrame for 'low' capslock use
low_capslock_df.head()
```

Out[189]:

	pseudID	text_clean	sum_senti	caps_cat
2	12	you can always google tutorials or feel free to ask me	0	low
3	14	Wizards here??? Amazing giveaway with We are giving away 10 WL and 10 OG spots!▯ 1▯Follow both Twi…	4	low
5	17	Heinrich Lossow painted this in 1880. You couldn’t Google no references, and it wasn’t no procreate. This took weeks, may…	0	low
6	22	If you want to be part of ourt collection:1. Fill the form in the link below:2. RT+Like this…	0	low
7	23	Growth #monitoring and #promotion provide an opportunity to prevent #malnutrition before it occurs. The funded…	0	low

In [188…
```
# Filtering rows in the DataFrame where the 'caps_cat' column has the value 'medium'
medium_capslock_df = categorized_df[categorized_df['caps_cat'] == 'medium']

# Displaying the first few rows of the filtered DataFrame for 'medium' capslock use
medium_capslock_df.head()
```

Out[188]:

	pseudID	text_clean	sum_senti	caps_cat
0	4	[EN-LIVE RECAP ▯]230203 “In MANILA”▯: 230204 “Tired and Sleepy Mildangz”▯:	0	medium
1	10	T-minus 7▯ days! Are you ready BLUR Community?blur list $BLUR on pancakeswap#OKX #Huobi #Bybit #Gateio #Bitget #Bi…	-1	medium
9	36	Apply for ApeList NOW! ▯GO ▯ FCFS : 777 WL ▯	Follow, Like, RT#Canto #CantoNFTs #Cant…	2	medium
10	42	You missed $CBONK airdrop? Dont worry,you are still early to join $APEAI Airdrop.How to join $ApeAI Airdrop:…	-2	medium
13	54	Buy Google Reviews	0	medium

In [187…
```
# Filtering rows in the DataFrame where the 'caps_cat' column has the value 'high'
high_capslock_df = categorized_df[categorized_df['caps_cat'] == 'high']

# Displaying the first few rows of the filtered DataFrame for 'high' capslock use
high_capslock_df.head()
```

Out[187]:

	pseudID	text_clean	sum_senti	caps_cat
4	15	Project for TENLEE TEN THAILAND x TENzenter7 YEARS 10GETHERDONATION SUPPORT LET'S LIGHTE…	2	high
12	50	Project for TENLEE TEN THAILAND x TENzenter7 YEARS 10GETHERDONATION SUPPORT LET'S LIGHTEN T…	2	high
21	75	Project for TENLEE TEN THAILAND x TENzenter7 YEARS 10GETHERDONATION SUPPORT LET'S LIGHTE…	2	high
28	97	#4KsooGOs WTS / LFB / PH GO WayV FANMEETING - BEYOND LIVE EVENT: LIVE STREAMING TICKET + OFFICIAL MD ✨Normal ETAFeb. 11 MD releasePRICES ARE ALL IN + TO FOLLOW LSF ✓ DOO & DOP: UNTIL FEB. 11, 2023 12PM only! ORDER FORM ⤵	2	high
48	168	2/7/2023 AM: TRAFFIC ACCIDENT NO INJURY at DEANS BRIDGE RD AND WHEELESS RD AUGUSTA GA	-2	high

Question 6. Calculating the reliability of the sentiment analysis

Since the categories are based on counts or percentages, I trust that they are reliable without manual verification of every uppercase letter.

Nevertheless, I will manually annotate the sentiment score according to the question's structure.

A. Exporting a random sample of at least 50 tweets to Excel:

Note: make sure that the `openpyxl` library is imported. If not installed, it can be done through a pip.

The initial two lines of code sample 50 tweets and save them to an Excel file in the same directory using the `.sample()` and `.to_excel()` methods. These lines are commented out with a `#` so that the code cell can be re-run without altering the `excel_50` DataFrame.

The randomly sampled DataFrame `excel_50` was saved to the notebook's directory. To maintain the output in this section, the initially sampled and saved DataFrame is loaded into the environment again using the `read_excel` function.

This DataFrame is not yet annotated. If someone wants to re-run this section, they need to remove the `#` from the first lines of code and then manually annotate the randomly generated tweets themselves. As a result, reliability scores may differ from the original annotations.

The code utilized in this section is derived from one of my own notebooks from the in-class exercise of DA4 Weekly Challenges, where we were tasked with a similar intercoder reliability task.

In [158…

```
# Installs the openpyxl package using pip
!pip install openpyxl
```

```
Requirement already satisfied: openpyxl in /Users/helgegeurtjacobusmoes/anaconda3/lib/py
thon3.10/site-packages (3.0.10)
Requirement already satisfied: et_xmlfile in /Users/helgegeurtjacobusmoes/anaconda3/lib/
python3.10/site-packages (from openpyxl) (1.1.0)
```

In [155…

```
# Importing the openpyxl library
import openpyxl

# Sampling 50 random rows from the DataFrame categorized_df and storing them in the vari
excel_50 = categorized_df.sample(50)

# Exporting the sampled data to an Excel file named "manual_coding.xlsx"
```

```
excel_50.to_excel("manual_coding.xlsx")

# This line of code is commented out and is intended to be run if the Kernel is restarte
# It reads the data from the Excel file "manual_coding.xlsx" back into the DataFrame exc
excel_50 = pd.read_excel("manual_coding.xlsx")
```

B. Manually coding the random sample for sentiment for the category:

The manual coding of this random sample for sentiment in the excel is coded in "man_cod". This is added in the submission. However, anyone with this notebook can run the code themselves. The given scores are the following:

- `-5` = extremely negative
- `-4` = very negative
- `-3` = medium negative
- `-2` = somewhat negative
- `-1` = slightly negative
- `0` = neutral
- `1` = slightly positive
- `2` = somewhat positive
- `3` = medium positive
- `4` = very positive
- `5` = extremely positive

After coding, I save the file as `manual_coding_completed.xlsx` to preserve the original, unannotated dataset. Subsequently, I load the annotated dataset into pandas. This DataFrame should be identical to the `excel_50` DataFrame, but with an additional column containing my annotated scores.

C. Loading the manually coded data on Pandas:

In this section, the manual coding is portrayed of' `manual_coding_completed.xlsx`.

In [157…
```
# Reading the Excel file "manual_coding_done.xlsx" and storing the data in the DataFrame
excel_annotated = pd.read_excel("manual_coding_completed.xlsx")

# Displaying the first few rows of the DataFrame excel_annotated
excel_annotated.head()
```

Out[157]:

	Unnamed: 0	pseudID	text_clean	sum_senti	caps_cat	man_cod
0	5591	16315	The tech companies are ALL IN on #ai Incredible how much activity there’s been the past few weeks.Big announcements fr…	0	low	0
1	6725	19525	Google Search People Cards Now Visible In US	0	medium	0
2	2010	6351	Google answers ChatGPT with its own chatbot	0	medium	0
3	2422	7893	absolutely, bc you should not expect people to expend mental labor for free when google is right there.	0	low	0
4	7133	20747	#WeCareKittyNFT Pre-Mint OFFER .049 eth and Limited 1000 Qty✅ Benefits ⬇ Fill the form…	0	medium	0

D. Merging the manual coding of the automated categorization:

Furthermore, I merge the two data frames based on the four columns that they share: `pseudID`, `text_clean`, `sum_senti` and `caps_cat`. In order to examine if they are compatible for joining, I use the `len()` function.

In [192…
```
# Printing the length of the initial data frame excel_50
print("Initial data frame length: ", len(excel_50))

# Merging excel_50 and excel_annotated data frames on the common columns pseudID, text_c
reliability_df = excel_50.merge(excel_annotated, on=['pseudID', 'text_clean', 'sum_senti

# Printing the length of the merged data frame reliability_df
print("Length of the merged data frame: ", len(reliability_df))
```

```
Initial data frame length:  50
Length of the merged data frame:  50
```

In [160…
```
# Inspecting the columns of the reliability data frame
reliability_df.columns
```

Out[160]:
```
Index(['Unnamed: 0_x', 'pseudID', 'text_clean', 'sum_senti', 'caps_cat',
       'Unnamed: 0_y', 'man_cod'],
      dtype='object')
```

Based on the inspection, the data frames portray an unnecessary index column (`Unnamed: 0`) during export/import from Excel.

Consequently, I will only choose relevant columns for calculating the reliability score through slicing and leaving this particular column out.

In [161…
```
# Selecting only the relevant columns pseudID, text_clean, sum_senti, and annotated_scor
reliability_df = reliability_df[["pseudID", "text_clean", "sum_senti", "man_cod"]]

# Displaying the first few rows of the updated DataFrame reliability_df
reliability_df.head()
```

Out[161]:

	pseudID	text_clean	sum_senti	man_cod
0	16315	The tech companies are ALL IN on #ai Incredible how much activity there's been the past few weeks.Big announcements fr…	0	0
1	19525	Google Search People Cards Now Visible In US	0	0
2	6351	Google answers ChatGPT with its own chatbot	0	0
3	7893	absolutely, bc you should not expect people to expend mental labor for free when google is right there.	0	0
4	20747	#WeCareKittyNFT Pre-Mint OFFER .049 eth and Limited 1000 Qty✅ Benefits ⬇ Fill the form…	0	0

E. Calculating intercoder reliability between the coding and the automated categorization:

In order to run a reliability test, it is important to install the `krippendorff` library.

When this is installed, the intercoder reliability can be calculated between the manual coding and the automated categorization. This code was retrieved from 13_1_Reliability.

In [164…
```
# Installing the krippendorff package using pip
! pip install krippendorff
```

```
Requirement already satisfied: krippendorff in /Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (0.6.1)
Requirement already satisfied: numpy<2.0,>=1.21 in /Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from krippendorff) (1.24.3)
```

In [165…]
```python
# Importing the krippendorff package
import krippendorff

# Calculating Krippendorff's alpha for the variables sum_senti and man_cod in the DataFr
# level_of_measurement parameter is set to "ordinal" indicating that the variables are o
krippendorff.alpha((reliability_df["sum_senti"], reliability_df["man_cod"]), level_of_me
```

Out[165]:
```
0.7528421549706401
```

The Krippendorff's alpha score of 0.753 indicates a moderate level of agreement. This score suggests that there is a notable level of intercoder reliability beyond what would be expected by chance, but there may still be room for improvement in the reliability of the ratings.

Note: This may also be due to the fact that some tweets are not fully transferred in the dataset and are cut off with `...`. For this analysis, I left these tweets in and considered this test as a reliable intercoder reliability test.

Question 7. Visualizing and providing descriptive statistics for the Research Question

In this section, I shall follow the instructions of this assignment further and attempt to answer the research question:

Does the use of CAPSLOCK correlate with the sentiment of the tweet?

A. Showing descriptive statistics for the IV (categorical variable) and the DV (sentiment):

In order to create visualizations the library `seaborn` is imported and `mathplotlib` is imported to edit the format of the figures. This has been retrieved from 12_0_DataPreparationVisualisation.

In [166…]
```python
# Importing the seaborn and matplotlib.pyplot libraries for visualization
import seaborn as sns
import matplotlib.pyplot as plt

# Setting the display format for float values to three decimal places
pd.set_option('display.float_format', lambda x: '%.3f' % x)

# Ensuring that plots are displayed inline in the Jupyter Notebook
%matplotlib inline
```

For the capslock categories (IV) the `.value_counts()` method is used to count the occurences of each unique value in the `caps_cat` column of the data frame, which is retrieved from 12_0_DataPreparationVisualisation.

In [167…]
```python
# Counting the occurrences of each category in the "caps_cat" column of the DataFrame ca
categorized_df["caps_cat"].value_counts()
```

Out[167]:
```
caps_cat
low       4608
medium    3604
high       199
Name: count, dtype: int64
```

For the sentiment score (DV) the `describe()` method is used in order to generate descriptive statistics for the `sum_senti` column of the data frame, which is retrieved from 12_0_DataPreparationVisualisation.

In [168…
```
# Generating descriptive statistics for the "sum_senti" column of the DataFrame categori
categorized_df["sum_senti"].describe()
```

Out[168]:
```
count   8411.000
mean       0.413
std        1.638
min       -5.000
25%        0.000
50%        0.000
75%        2.000
max        5.000
Name: sum_senti, dtype: float64
```

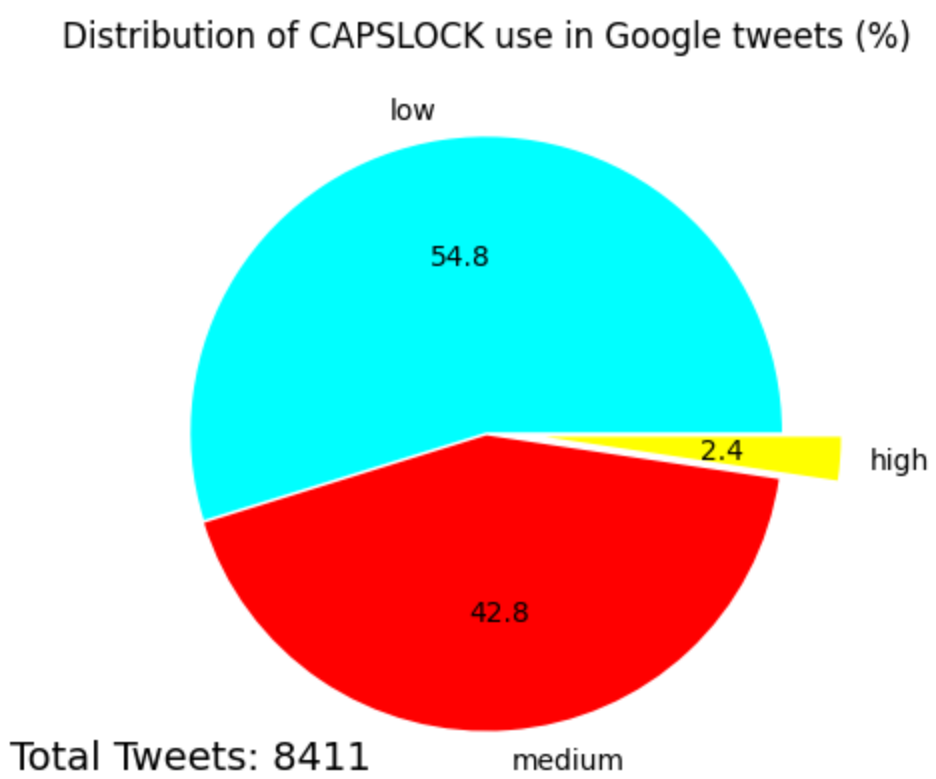
B. Creating a univariate visualisation for the IV:

For the independent variable, I decided to use a pie chart, because the `high` count seemed to be significantly lower than the other categorizations. Therefore, this visualization seemed to be the best fit in order to portray this information. It shows the huge difference in `high` distribution in contrast with `low` and `medium`.

Therefore, I used the `plot.pie()` method, which was retrieved from the GitHub of the Big Data and Automated Content Analysis course: Visualization. Besides this GitHub, I also resorted to the Matplotlib for further instructions to fine tune the graph.

In [191…
```
# Generating a pie chart to visualize the distribution of categories in the "caps_cat" c
categorized_df['caps_cat'].value_counts().plot.pie(
    autopct='%1.1f',  # Display percentage values with one decimal place
    title='Distribution of CAPSLOCK use in Google tweets (%)',  # Title of the pie chart
    colors=['cyan', 'red', 'yellow'],  # Colors for each category
    explode=[0, .01, .2]  # Explode the slices to emphasize certain categories
)

# Adding text to display the total number of tweets in the center of the plot
plt.text(-1, -1.1, f'Total Tweets: {len(categorized_df)}', ha='center', va='center', fon

# Removing the y-label
plt.ylabel('')
```

Out[191]:
```
Text(0, 0.5, '')
```



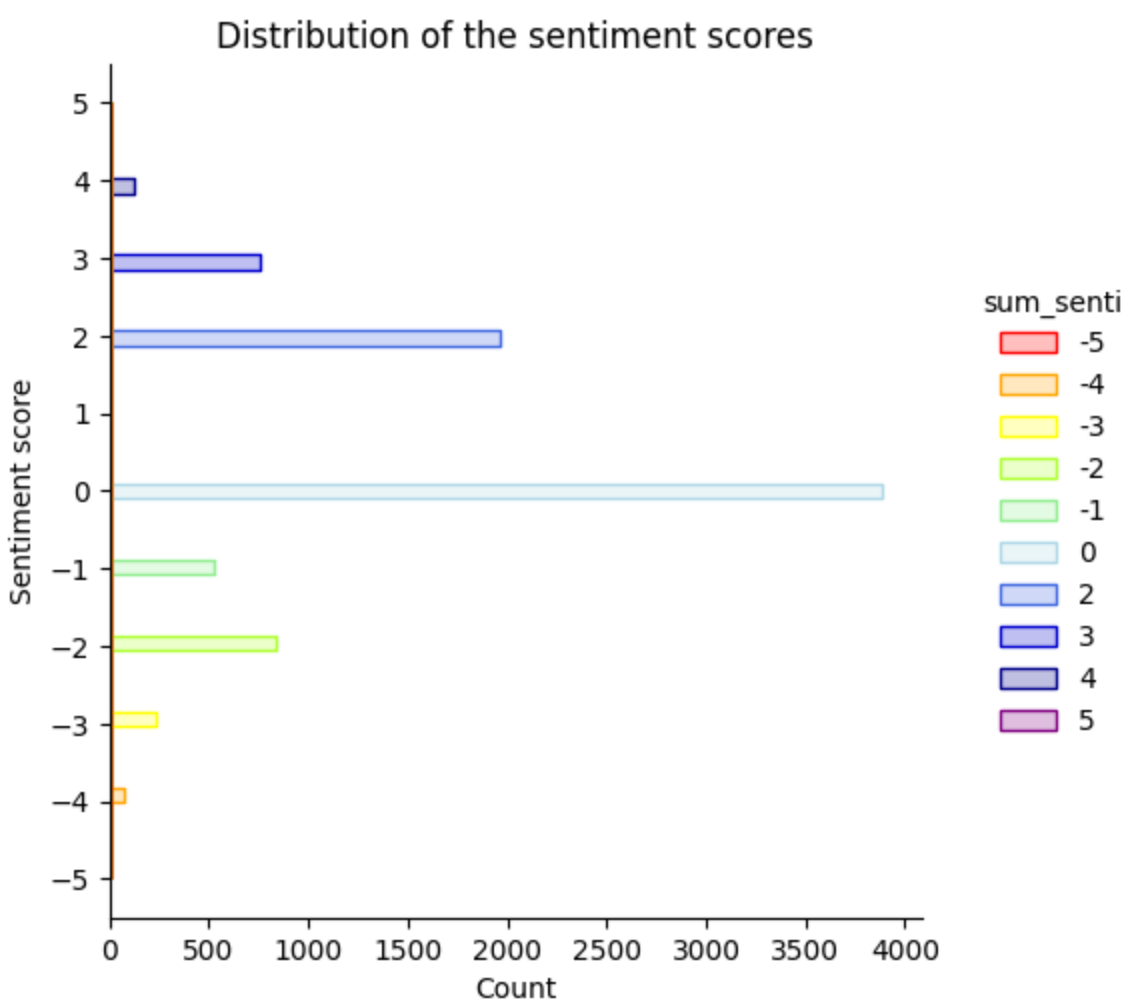
C. Creating a univariate visualisation for the DV:

For this visualization, I decided to use a histogram in order to portray the different ticks to show the difference in the presence of the multiple sentiment scores ranging from `-5` as 'very negative' to `5` as 'very positive'.

Besides 12_0_DataPreparationVisualisation, I resorted to Matplotlib and Seaborn for the visualization of the data in this plot.

In [182…

```
# Defining a color palette for each sentiment score
colors = { -5: 'red', -4: 'orange', -3: 'yellow', -2: 'greenyellow', -1: 'lightgreen',
           0: 'lightblue', 1: 'deepskyblue', 2: 'royalblue', 3: 'mediumblue',
           4: 'darkblue', 5: 'purple'}

# Defining the plot using the color palette
sns.displot(data=categorized_df, y="sum_senti", hue="sum_senti", palette=colors, element

# Setting the title of the plot
plt.title("Distribution of the sentiment scores")

# Showing each tick in the y-axis
plt.yticks([-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5])

# Editing label on the y axis
plt.ylabel("Sentiment score")

# Displaying the figure
plt.show()
```



D. Creating a bivariate visualisation with the IV and DV in the same chart:

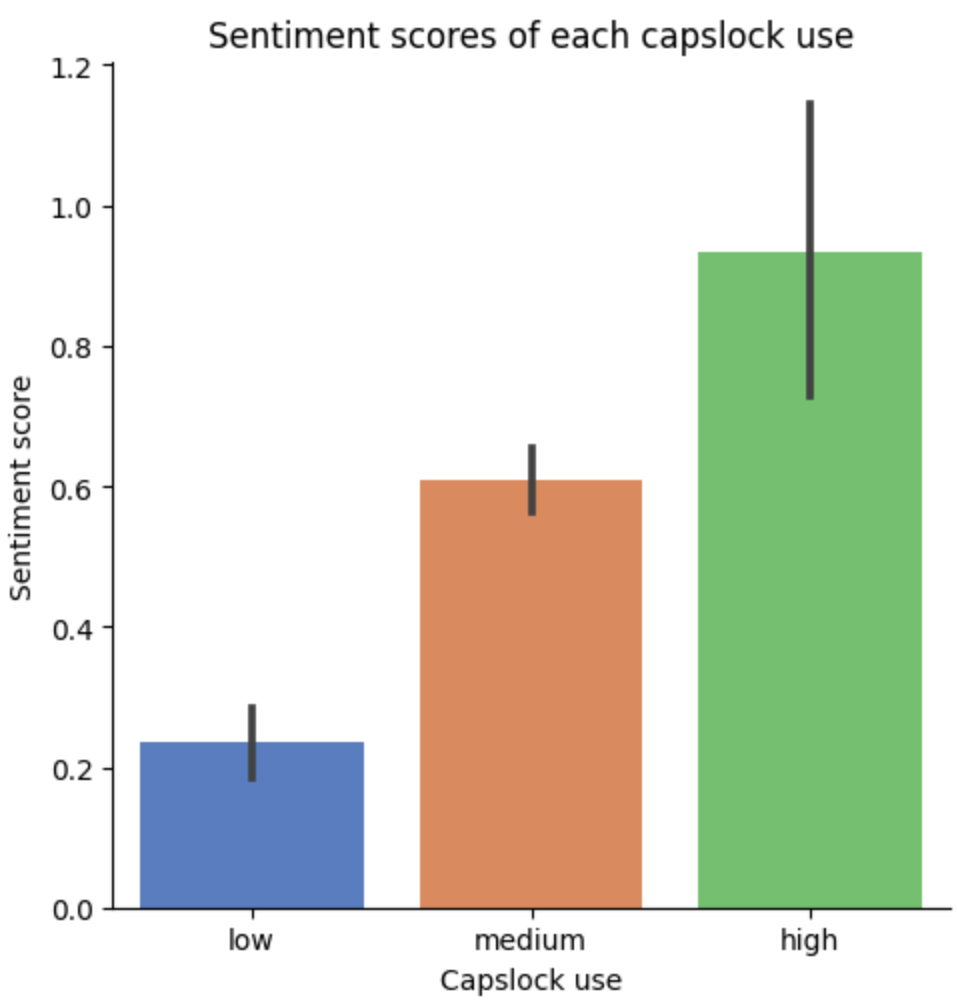
In order to create a bivariate visualization of a categorical variable and a continuous variable, I decided to use a barplot, because it allows for a comparison in the distribution of the continuous variable across different categories, which makes it possible to measure the average sentiment score for each category. The code for this visualization was retrieved from Seaborn.

In [194…

```
# Creating a categorical plot (catplot) using seaborn
sns.catplot(
    data=categorized_df,         # Dataframe to use for plotting
    x="caps_cat",            # Categorical variable to be displayed on the x-axis
    y="sum_senti",               # Continuous variable to be displayed on the y-axis
    kind="bar",                  # Type of plot (bar plot)
    order=["low", "medium", "high"],  # Order of categories on the x-axis
    palette="muted"             # Color palette for the bars
)

# Setting the title of the plot
plt.title("Sentiment scores of each capslock use")

# Labeling the x-axis
plt.xlabel("Capslock use")

# Labeling the y-axis
plt.ylabel("Sentiment score")
```

Out[194]:

```
Text(9.444444444444445, 0.5, 'Sentiment score')
```



E. Showing descriptive statistics of the DV grouped by the IV including interpretation:

In order to display the descriptives of the sentiment scores that are grouped by the category of capslock use, I adjusted code that was used in 12_0_DataPreparationVisualisation.

In [190…
```
# Group the DataFrame by the 'caps_cat' column and calculate descriptive statistics for
categorized_df.groupby(['caps_cat'])['sum_senti'].describe()
```

Out[190]:

caps_cat	count	mean	std	min	25%	50%	75%	max
high	199.000	0.935	1.498	-4.000	0.000	2.000	2.000	4.000
low	4608.000	0.236	1.772	-5.000	-1.000	0.000	2.000	5.000
medium	3604.000	0.610	1.426	-4.000	0.000	0.000	2.000	5.000

Interpretation:

The majority of tweets exhibit either medium or low capslock usage. Across all categories, sentiment scores tend to be more positive than negative. A linear trend can be observed, since the average sentiment score becomes more positive and the number of uppercase characters increases. Based on this exploration, we can deduce that capslock is predominantly employed to convey positive sentiments rather than negative ones. Therefore, answering the research question that the use of CAPSLOCK does correlate with the sentiment of the tweet.

Interpretation

Explaining your process

Write a brief description explaining how you created or assembled the code (i.e., sources used, and how did you adjust the code). In this description, include a brief reflection of the main challenges you had during the process, and how you handled these challenges.

Wordcount: maximum of 200 words. Please inform the wordcount for this answer

Answer:

For me, coding starts with understanding what the objective is. In this case, it is to explore the use of capslock in tweets on sentiment. To realize this, I primarily utilized resources from the course's GitHub repository, solutions of the Weekly Challenges, the online book authored by van Arila et al. (2022), and relied on internet searches for specific processes and function usage.

I found it difficult to conceive a challenge and finding two datasets in the One Drive that would be interesting to explore. To get inspiration, I would go on YouTube and look for data projects that have done sentiment analyses.

In contrast to challenge 1, much of the process involved trial-and-error. Consequently, I would frequently verify the code's behavior using functions like `print()` and `head()` to ensure it aligned with my intentions and the objective. I also found the "Restart & Run All" option helpful for maintaining the correct cell order and observing whether any alterations in the data are reflected in later cells.

Furthermore, finding the suitable visualization seemed to be more challenging than initially thought. During this process, I would keep in mind what was taught in DA4. I became more critical of my plots and whether it told the story of the results effectively. In my opinion, a graph needs little to no explanation and the reader needs to be able to understand the results purely from the plot. Based on my visualizations, I feel that my attempts were successful.

Word count: 183

Explaining your code

You used a lot of code above to answer the seven questions. Select what you consider to be the three most important steps in the code, justify why they are important, and explain the commands being used.

Note: there are different correct answers to what the three most important steps of the code are. We want to understand your reasoning/justification for this, and will accept different answers depending on the logic and clarity of your argumentation.

Wordcount: maximum of 200 words. Please inform the wordcount for this answer

Answer:

Firstly, an essential method utilized in this particular challenge is `.apply()`, especially in: `final_df = merged_df[["positive", "negative", "neutral"]].apply(pd.to_numeric)`. This function facilitates the application of a method that changes a data type (in this case numeric) across an entire data c columns. Without this specific code, the numeric remained being presented as strings or

another non-numeric format, which would make answering the questions after Question 4 considerably more challenging.

Secondly, to explore the sentiment in the tweets, an additional dataset containing the sentiments needed to be merged with the Google dataset. Therefore, aligning the data types, lengths of tweets and sentiment scores is crucial for effective dataset merging. Verifying the lengths of datasets prevents data loss during merging, and examining column data types ensures compatibility. Additionally, performing the merge operation with an inner join ensures the intersection of keys from both frames using the `merge()` function. Checking the resulting `merged_df` length confirms successful merging, which was expected to contain the total number of tweets: 8411.

Lastly, an important step in the code involves pseudonimizing the data by removing all original `id` values. This ensures data privacy and protect sensitive information involving personal identifiable data. To enhance privacy, the `text` column is pseudonimized by replacing Twitter mentions (strings starting with '@') with '@user'. This is achieved using regular expressions and the `.replace()` method from 11_DataPrivacy. Moreover `.reset_index()` and `rename()` were crucial to assign new `id` values to each tweet to make them unidentifiable.

Word count: 193

Quality assurance

You are assembling code from different sources - some of it you learned in the tutorial, you may have reused code we provided, and in some cases you may have used different online sources. But how do you know that the code actually worked and delivered the expected results? Please explain how someone else can verify that the code worked (e.g., what should they look at) and which steps you built into the code for yourself to know that it worked (explicitly indicate at least three steps)

Wordcount: maximum of 200 words. Please inform the wordcount for this answer

Answer:

In general, if someone else wants to replicate or verify whether the code in this Notebook worked, I would advice to avoid hardcoding values. For instance, refrain from specifying system-specific file paths directly within functions like `pd.read_` or `pd.to_`, which load and save files to the working directory (where the Markdown file resides). Additionally, maintaining the order of cells is fundamental. After making changes, it is important to always save them and rerun the entire Kernel to prevent referencing undefined variables or data frames resulting from cell rearrangements. Moreover, when certain code commands require additional libraries not installed by default, such as `krippendorff`, it is important to install these libraries. Lastly, I would recommend to ensure that all relevant lines of code are properly annotated either using markdown or in-line comments `#` to enhance readability and clarity.

For this specific challenge, it is also worth noting that checking the code manually is important, since some tweets are not fully transferred in the datasets that were used in this challenge, which possibly affected the intercoder reliability and outcome of the analysis. Consequently, the Krippendorff's alpha score of 0.753 suggests a moderate level of agreement among the coders. While this indicates a notable level of intercoder reliability beyond chance, there is still room for improvement in the ratings' reliability. Besides the Krippendorff's alpha, the merging of the datasets were an integral aspect of this challenge. Merging can

lead to missing values, which could effect the data output. Therefore, checking the quality of the merging with `missing_values = merged_df.isna().sum()` confirms a successful merging.

Word count: 189

Reflection

You have used a real-life dataset that contains information on tweets posted by real Twitter users. In this reflection, we ask you to focus on ethical and privacy considerations stemming from using such data for digital analytics. Please discuss the privacy risks as introduced by Tucker (2019) involved when such data from Twitter users are used for digital analytics (for example, answering your research questions), and what data-related considerations - related to and beyond privacy - need to be taken into account when collecting and analyzing tweets (following the Data Ethics Decision Aid).

Wordcount: maximum 500 words. Please inform the wordcount in this section.

Answer:

Social media data presents more information than we initially perceive. Although publicly available, data that can be retrieved from comments and likes can hold ethical dilemmas, including concerns about privacy and informed consent. When users decide to use social media, they never assume that their information, however trivial it may seem, might be used against their favor by third parties. Therefore, the user should be aware of the fact that once the data is created, it may outlive the creator, which may lead to unforeseen repercussions (Tucker, 2019).

This “data persistence” (Tucker, 2019, p. 426) implies that social media data, which has been collected for one purpose, might be repurposed for unintended uses, posing risks to individuals whose data is involved. Despite the fact that researchers may have good intentions, careless handling of data can result in privacy breaches or biased treatment by classification algorithms (Tucker, 2019). In the case of this challenge, I collected data for a sentiment analysis for this specific challenge. However, without properly anonymizing and pseudonymizing the data, it could potentially be repurposed to infer information about individuals based on their tweets, possibly associating them with unintended groups or worse: causing harm and (online) abuse when published. This is referred to as "data repurposing" (Tucker, 2019, p. 423).

Furthermore, it comes as no surprise that social media companies recognize the potential of the social media data for their business value (Culnan et al., 2010). Consequently, these companies leverage user data and metadata to analyze digital footprints for their own benefit. This entails extracting: demographic attributes and political orientation without users being aware of this (Hinds & Joinson, 2018). An example of how data has been repurposed is the Scandal of Cambridge Analytica in 2018 (Hinds et al., 2020). It was revealed that the political consulting firm collected personal data from millions of Facebook users without their consent. This data was then used for political advertising during the elections.

Expanding on repurposing data, Tucker (2019) also highlights the concept of "data spillovers" (p. 423), where analysts using datasets may accidentally expose identifiable information, such as `id` or `author_id`. This potentially leads to traceability, privacy breaches or other negative consequences for users. Therefore, researchers must adhere to the DEDA Worksheet and exercise caution throughout the analysis process (Franzke, Muis, & Schäfer, 2021). The Data Ethics Decision Aid framework is specifically

designed to moderate and advance these responsible data practices as well as creating awareness of ethical issues (Franzke, Muis, & Schäfer, 2021).

In conclusion, Tucker's (2019) highlighting of privacy concerns, particularly the potential risk of compromising individual privacy through predictive models, not only emphasizes the necessity of treating information with great care but also creates an awareness for the user to be mindful of what is presented on social media. The key of using this valuable resource is to balance between transparency and privacy to the extent that the individuals' information is safeguarded. Since the harm caused by these mistakes are irreversible, it is integral to remain cautious throughout the data analysis process.

Word count: 498

Bibliography

Arcila, W. van A., Damian Trilling &. Carlos. (2022, March 11). Computational Analysis of Communication. https://cssbook.net/

Culnan, M., McHugh, P., & Zubillaga, J. (2010). How Large U.S. Companies Can Use Twitter and Other Social Media to Gain Business Value. MIS Quarterly Executive, 9, 243–259.

Franzke, A. S., Muis, I., & Schäfer, M. T. (2021). Data Ethics Decision Aid (DEDA): A dialogical framework for ethical inquiry of AI and data projects in the Netherlands. *Ethics and Information Technology, 23*(3), 551–567. https://doi.org/10.1007/s10676-020-09577-5

Hinds, J., & Joinson, A. N. (2018). What demographic attributes do our digital footprints reveal? A systematic review. *PLOS* ONE, 13(11), e0207112. https://doi.org/10.1371/journal.pone.0207112

Hinds, J., Williams, E. J., & Joinson, A. N. (2020). “It wouldn’t happen to me”: Privacy concerns and perspectives following the Cambridge Analytica scandal. International Journal of Human-Computer Studies, 143, 102498. https://doi.org/10.1016/j.ijhcs.2020.102498

Python Pandas—Options and Customization. (n.d.). Retrieved March 4, 2024, from https://www.tutorialspoint.com/python_pandas/python_pandas_options_and_customization.htm

Quick start guide—Matplotlib 3.8.3 documentation. (n.d.). Retrieved March 6, 2024, from https://matplotlib.org/stable/users/explain/quick_start.html#parts-of-a-figure

seaborn.histplot—Seaborn 0.13.2 documentation. (n.d.). Retrieved March 6, 2024, from https://seaborn.pydata.org/generated/seaborn.histplot.html#seaborn.histplot

Topic Modeling and Sentiment Analysis on Tweets. (n.d.). Laminar Insight. Retrieved March 4, 2024, from https://www.laminarinsight.com/post/twitter-topic-modeling-sentiment-analysis/

Tucker, C. (2019). Privacy, Algorithms, and Artificial Intelligence. In A. Agrawal, J. Gans, & A. Goldfarb (Eds.), *The Economics of Artificial Intelligence: An Agenda* (pp. 423–437). University of Chicago Press.