MIE451/1513 Decision Support System -- Data Science Solution

Be sure to let us know:

i. what location you chose (and remember to sign up on Piazza so there are no duplicates),

Cornwall, Ontario

ii. what preprocessing steps you implemented

Just the recommended filtering step on Piazza for ensuring all the hotels are from Cornwall, Ontario

```
# importing useful libraries

#Plot
import matplotlib.pyplot as plt
import seaborn as sns
#%matplotlib inline
from wordcloud import WordCloud

#Data Packages
import math
import pandas as pd
import numpy as np

#Progress bar
from tqdm import tqdm

#Counter
from collections import Counter

#Operation
import operator

#Natural Language Processing Packages
import re
import nltk

## Download Resources
nltk.download("vader_lexicon")
nltk.download("stopwords")
nltk.download("averaged_perceptron_tagger")
nltk.download("wordnet")

from nltk.sentiment import SentimentAnalyzer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.sentiment.util import *
from nltk import tokenize
from nltk.corpus import stopwords
from nltk.tag import PerceptronTagger
from nltk.data import find

## Machine Learning
import sklearn
import sklearn.metrics as metrics

## Data Visualization
import folium
from tabulate import tabulate
from scipy.stats.kde import gaussian_kde

## Geolocation
import geopy
from geopy.geocoders import Photon
from geopy.extra.rate_limiter import RateLimiter
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\kirby\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\kirby\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\kirby\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\kirby\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
%matplotlib inline
# setting pandas' display
pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('max_colwidth', 500)
```

Q1

```
df = pd.read_csv('reviews.csv', header=None)
df.columns = ['filePath','hotelName','reviewColumn','ratingScore','groundTruth',
              'date_stamp','streetAddress', 'City',
              'Province','postalCode']
#['path','hotel','review','score','bool','date','address','city','province','postal']
```

```
df.head()
```

Please follow our blog to see more information about new features, tips and tricks, and featured notebooks such as Analyzing a Bank Failure with Colab.

2023-11-08

- Launched Secrets, for safe storage of private keys on Colab (tweet)
- Fixed issue where TensorBoard would not load (#3990)
- Python package upgrades
  - lightgbm 4.0.0 -> 4.1.0
  - bigframes 0.10.0 -> 0.12.0
  - bokeh 3.2.2 -> 3.3.0
  - duckdb 0.8.1 -> 0.9.1
  - numba 0.56.4 -> 0.58.1
  - tweepy 4.13.0 -> 4.14.0
  - jax 0.4.16 -> 0.4.20
  - jaxlib 0.4.16 -> 0.4.20

2023-10-23

- Updated the **Open notebook** dialog for better usability and support for smaller screen sizes
- Added smart paste support for data from Google Sheets for R notebooks
- Enabled showing release notes in a tab
- Launched AI coding features for Pro/Pro+ users in Australia AU Canada CA India IN and Japan JP (tweet)
- Python package upgrades
  - earthengine-api 0.1.357 -> 0.1.375
  - flax 0.7.2 -> 0.7.4
  - geemap 0.27.4 -> 0.28.2
  - jax 0.4.14 -> 0.4.16
  - jaxlib 0.4.14 -> 0.4.16
  - keras 2.13.1 -> 2.14.0
  - tensorboard 2.13.0 -> 2.14.1
  - tensorflow 2.13.0 -> 2.14.0
  - tensorflow-gcs-config 2.13.0 -> 2.14.0
  - tensorflow-hub 0.14.0 -> 0.15.0
  - tensorflow-probability 0.20.1 -> 0.22.0
  - torch 2.0.1 -> 2.1.0
  - torchaudio 2.0.2 -> 2.1.0
  - torchtext 0.15.2 -> 0.16.0
  - torchvision 0.15.2 -> 0.16.0
  - xgboost 1.7.6 -> 2.0.0
- Python package inclusions
  - bigframes 0.10.0
  - malloy 2023.1056

2023-09-22

- Added the ability to scope an AI generated suggestion to a specific Pandas dataframe (tweet)
- Added Colab link previews to Docs (tweet)
- Added smart paste support for data from Google Sheets
- Increased font size of dropdowns in interactive forms
- Improved rendering of the notebook when printing
- Python package upgrades
  - tensorflow 2.12.0 -> 2.13.0
  - tensorboard 2.12.3 -> 2.13.0
  - keras 2.12.0 -> 2.13.1
  - tensorflow-gcs-config 2.12.0 -> 2.13.
  - scipy 1.10.1-> 1.11.2
  - cython 0.29.6 -> 3.0.2
- Python package inclusions
  - geemap 0.26.0

2023-08-18

- Added "Change runtime type" to the menu in the connection button
- Improved auto-reconnection to an already running notebook (#3764)
- Increased the specs of our highmem machines for Pro users
- Fixed `add-apt-repository` command on Ubuntu 22.04 runtime (#3867)
- Python package upgrades
  - bokeh 2.4.3 -> 3.2.2
  - cmake 3.25.2 -> 3.27.2
  - cryptography 3.4.8 -> 41.0.3

	filePath	hotelName	reviewColumn	ratingScore
			"Should have known better . upon checking in was told that service at Christmas \u002F Boxing Day would be minimal. DID not know that meant NON existent. We never got room clean up \u002F towel refresh or any service at all	

```
# Use vader to evaluated sentiment of reviews
def evalSentences(sentences, to_df=False, columns=[]):
    #Instantiate an instance to access SentimentIntensityAnalyzer class
    sid = SentimentIntensityAnalyzer()
    pdlist = []
    if to_df:
        for sentence in tqdm(sentences):
            ss = sid.polarity_scores(sentence)
            pdlist.append([sentence]+[ss['compound']])
        reviewDf = pd.DataFrame(pdlist)
        reviewDf.columns = columns
        return reviewDf

    else:
        for sentence in tqdm(sentences):
            print(sentence)
            ss = sid.polarity_scores(sentence)
            for k in sorted(ss):
                print('{0}: {1}, '.format(k, ss[k]), end='')
            print()
```

PEI back to our

```
reviewDF = evalSentences(df['reviewColumn'].values, to_df=True, columns=['reviewCol','va
```

```
  6%|▌         | 180/2889 [00:00<00:01, 1751.68it/s]100%|██████████| 2889/2889 [00
```

tub.

```
df['vader'] = reviewDF['vader']
```

tub however

```
df.groupby('hotelName', as_index=False)[['ratingScore','vader']].mean()
```

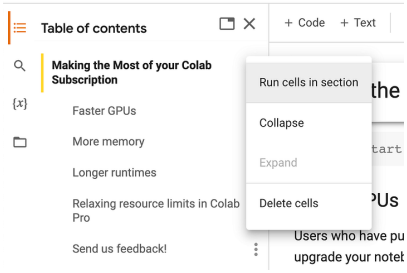
- dask 2022.12.1 -> 2023.8.0
- distributed 2022.12.1 -> 2023.8.0
- earthengine-api 0.1.358 -> 0.1.364
- flax 0.7.0 -> 0.7.2
- ipython-sql 0.4.0 -> 0.5.0
- jax 0.4.13 -> 0.4.14
- jaxlib 0.4.13 -> 0.4.14
- lightgbm 3.3.5 -> 4.0.0
- mkl 2019.0 -> 2023.2.0
- notebook 6.4.8 -> 6.5.5
- numpy 1.22.4 -> 1.23.5
- opencv-python 4.7.0.72 -> 4.8.0.76
- pillow 8.4.0 -> 9.4.0
- plotly 5.13.1 -> 5.15.0
- prettytable 0.7.2 -> 3.8.0
- pytensor 2.10.1 -> 2.14.2
- spacy 3.5.4 -> 3.6.1
- statsmodels 0.13.5 -> 0.14.0
- xarray 2022.12.0 -> 2023.7.0

- Python package inclusions
  - PyDrive2 1.6.3

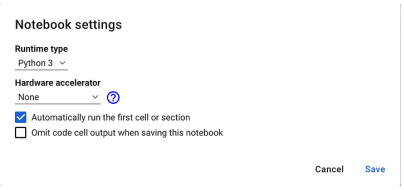
2023-07-21

- Launched auto-plotting for dataframes, available using the chart button that shows up alongside datatables (post)

- Added a menu to the table of contents to support running a section or collapsing/expanding sections (post)



- Added an option to automatically run the first cell or section, available under Edit -> Notebook settings (post)



- Launched Pro/Pro+ to Algeria, Argentina, Chile, Ecuador, Egypt, Ghana, Kenya, Malaysia, Nepal, Nigeria, Peru, Rwanda, Saudi Arabia, South Africa, Sri Lanka, Tunisia, and Ukraine (tweet)
- Added a command, "Toggle tab moves focus" for toggling tab trapping in the editor (Tools -> Command palette, "Toggle tab moves focus")
- Fixed issue where `files.upload()` was sometimes returning an incorrect filename (#1550)
- Fixed f-string syntax highlighting bug (#3802)
- Disabled ambiguous characters highlighting for commonly used LaTeX characters (#3648)
- Upgraded Ubuntu from 20.04 LTS to 22.04 LTS
- Updated the Colab Marketplace VM image
- Python package upgrades:
  - autograd 1.6.1 -> 1.6.2
  - drivefs 76.0 -> 77.0
  - flax 0.6.11 -> 0.7.0
  - earthengine-api 0.1.357 -> 0.1.358
  - GDAL 3.3.2->3.4.3
  - google-cloud-bigquery-storage 2.20.0 -> 2.22.2
  - gspread-dataframe 3.0.8 -> 3.3.1
  - holidays 0.27.1 -> 0.29
  - jax 0.4.10 -> jax 0.4.13
  - jaxlib 0.4.10 -> jax 0.4.13

	hotelName	ratingScore	vader

```
hotel_scores = df.groupby('hotelName', as_index=False)[['ratingScore','vader']].mean()
hotel_scores['vader_median'] = df.groupby('hotelName', as_index=False)['vader'].median()
hotel_scores.columns = ['hotelName','mean ground truth', 'mean vader compound', 'median
hotel_scores
```

	hotelName	mean ground truth	mean vader compound	median vader compound
0	Auberge Chesley's Inn	4.807018	0.935478	0.97300
1	Best Western Parkway Inn & Conference Centre	4.548885	0.874945	0.95730
2	Century Motel	4.233766	0.834825	0.93160
3	Comfort Inn	3.031034	0.574015	0.89950
4	Elect Inn 5	3.323077	0.529631	0.83600
5	First Canada Inns	4.015873	0.729245	0.92585
6	Howard Johnson by Wyndham Cornwall	1.789474	0.041847	0.03870
7	Indigo Inn	2.187500	0.201037	0.45765
8	Martin's Inn	2.666667	0.303600	0.37765
9	Monte Carlo	1.800000	-0.029906	-0.36370

```
true_sort = hotel_scores.sort_values('mean ground truth', ascending=False, ignore_inde
top_5_true = true_sort[:5]['hotelName']
bottom_5_true = true_sort[-5:]['hotelName']
true_sort
```

	hotelName	mean ground truth	mean vader compound	median vader compound
0	Auberge Chesley's Inn	4.807018	0.935478	0.97300
1	Best Western Parkway Inn & Conference Centre	4.548885	0.874945	0.95730
2	Century Motel	4.233766	0.834825	0.93160
3	First Canada Inns	4.015873	0.729245	0.92585
4	Ramada by Wyndham Cornwall	3.958763	0.727240	0.92645
5	Elect Inn 5	3.323077	0.529631	0.83600
6	Super 8 by Wyndham Cornwall ON	3.276364	0.476064	0.83920
7	Regency Inn & Suites	3.200000	0.786460	0.97800

```
vader_mean_sort = hotel_scores.sort_values('mean vader compound', ascending=False, ign
top_5_vader_mean = vader_mean_sort[:5]['hotelName']
bottom_5_vader_mean = vader_mean_sort[-5:]['hotelName']
vader_mean_sort
```

- jupyterlab-widgets 3.0.7 -> 3.0.8
- nbformat 5.9.0 -> 5.9.1
- opencv-python-headless 4.7.0.72 -> 4.8.0.74
- pygame 2.4.0 -> 2.5.0
- spacy 3.5.3 -> 3.5.4
- SQLAlchemy 2.0.16 -> 2.0.19
- tabulate 0.8.10 -> 0.9.0
- tensorflow-hub 0.13.0 -> 0.14.0

2023-06-23

- Launched AI coding features to subscribed users starting with Pro+ users in the US (tweet, post)
- Added the Kernel Selector in the Notebook Settings (tweet)
- Fixed double space trimming issue in markdown #3766
- Fixed run button indicator not always centered #3609
- Fixed inconsistencies for automatic indentation on multi-line #3697
- Upgraded Python from 3.10.11 to 3.10.12
- Python package updates:
  - duckdb 0.7.1 -> 0.8.1
  - earthengine-api 0.1.350 -> 0.1.357
  - flax 0.6.9 -> 0.6.11
  - google-cloud-bigquery 3.9.0 -> 3.10.0
  - google-cloud-bigquery-storage 2.19.1 -> 2.20.0
  - grpcio 1.54.0 -> 1.56.0
  - holidays 0.25 -> 0.27.1
  - nbformat 5.8.0 -> 5.9.0
  - prophet 1.1.3 -> 1.1.4
  - pydata-google-auth 1.7.0 -> 1.8.0
  - spacy 3.5.2 -> 3.5.3
  - tensorboard 2.12.2 -> 2.12.3
  - xgboost 1.7.5 -> 1.7.6
- Python package inclusions:
  - gcsfs 2023.6.0
  - geopandas 0.13.2
  - google-cloud-bigquery-connection 1.12.0
  - google-cloud-functions 1.13.0
  - grpc-google-iam-v1 0.12.6
  - multidict 6.0.4
  - tensorboard-data-server 0.7.1

2023-06-02

- Released the new site colab.google
- Published Colab's Docker runtime image to us-docker.pkg.dev/colab-images/public/runtime (tweet, instructions)
- Launched support for Google children accounts (tweet)
- Launched DagsHub integration (tweet, post)
- Upgraded to Monaco Editor Version 0.37.1
- Fixed various Vim keybinding bugs
- Fixed issue where the N and P letters sometimes couldn't be typed (#3664)
- Fixed rendering support for compositional inputs (#3660, #3679)
- Fixed lag in notebooks with lots of cells (#3676)
- Improved support for R by adding a Runtime type notebook setting (Edit -> Notebook settings)
- Improved documentation for connecting to a local runtime (Connect -> Connect to a local runtime)
- Python package updates:
  - holidays 0.23 -> 0.25
  - jax 0.4.8 -> 0.4.10
  - jaxlib 0.4.8 -> 0.4.10
  - pip 23.0.1 -> 23.1.2
  - tensorflow-probability 0.19.0 -> 0.20.1
  - torch 2.0.0 -> 2.0.1
  - torchaudio 2.0.1 -> 2.0.2
  - torchdata 0.6.0 -> 0.6.1
  - torchtext 0.15.1 -> 0.15.2
  - torchvision 0.15.1 -> 0.15.2
  - tornado 6.2 -> 6.3.1

2023-05-05

- Released GPU type selection for paid users, allowing them to choose a preferred NVidia GPU
- Upgraded R from 4.2.3 to 4.3.0
- Upgraded Python from 3.9.16 to 3.10.11
- Python package updates:
  - attrs 22.2.0 -> attrs 23.1.0

	hotelName	mean ground truth	mean vader compound	median vader compound
0	Auberge Chesley's Inn	4.807018	0.935478	0.97300
1	Best Western Parkway Inn & Conference Centre	4.548885	0.874945	0.95730
2	Century Motel	4.233766	0.834825	0.93160
3	Regency Inn & Suites	3.200000	0.786460	0.97800
4	First Canada Inns	4.015873	0.729245	0.92585
5	Ramada by Wyndham Cornwall	3.958763	0.727240	0.92645
6	Comfort Inn	3.031034	0.574015	0.89950

```
vader_median_sort = hotel_scores.sort_values('median vader compound', ascending=False,
top_5_vader_median = vader_median_sort[:5]['hotelName']
bottom_5_vader_median = vader_median_sort[-5:]['hotelName']
vader_median_sort
```

	hotelName	mean ground truth	mean vader compound	median vader compound
0	Regency Inn & Suites	3.200000	0.786460	0.97800
1	Auberge Chesley's Inn	4.807018	0.935478	0.97300
2	Best Western Parkway Inn & Conference Centre	4.548885	0.874945	0.95730
3	Century Motel	4.233766	0.834825	0.93160
4	Ramada by Wyndham Cornwall	3.958763	0.727240	0.92645
5	First Canada Inns	4.015873	0.729245	0.92585
6	Comfort Inn	3.031034	0.574015	0.89950
7	Super 8 by Wyndham Cornwall ON	3.276364	0.476064	0.83920

TOP 5

```
top5 = pd.concat([top_5_true,top_5_vader_mean,top_5_vader_median], axis=1)
top5.columns = ['mean ground truth','mean vader compound','median vader compound']
top5
```

	mean ground truth	mean vader compound	median vader compound
0	Auberge Chesley's Inn	Auberge Chesley's Inn	Regency Inn & Suites
1	Best Western Parkway Inn & Conference Centre	Best Western Parkway Inn & Conference Centre	Auberge Chesley's Inn
2	Century Motel	Century Motel	Best Western Parkway Inn & Conference Centre
3	First Canada Inns	Regency Inn & Suites	Century Motel

- earthengine-api 0.1.349 -> earthengine-api 0.1.350
- flax 0.6.8 -> 0.6.9
- grpcio 1.53.0 -> 1.54.0
- nbclient 0.7.3 -> 0.7.4
- tensorflow-datasets 4.8.3 -> 4.9.2
- termcolor 2.2.0 -> 2.3.0
- zict 2.2.0 -> 3.0.0

2023-04-14

- Python package updates:
  - google-api-python-client 2.70.0 -> 2.84.0
  - google-auth-oauthlib 0.4.6 -> 1.0.0
  - google-cloud-bigquery 3.4.2 -> 3.9.0
  - google-cloud-datastore 2.11.1 -> 2.15.1
  - google-cloud-firestore 2.7.3 -> 2.11.0
  - google-cloud-language 2.6.1 -> 2.9.1
  - google-cloud-storage 2.7.0 -> 2.8.0
  - google-cloud-translate 3.8.4 -> 3.11.1
  - networkx 3.0 -> 3.1
  - notebook 6.3.0 -> 6.4.8
  - jax 0.4.7 -> 0.4.8
  - pandas 1.4.4 -> 1.5.3
  - spacy 3.5.1 -> 3.5.2
  - SQLAlchemy 1.4.47 -> 2.0.9
  - xgboost 1.7.4 -> 1.7.5

2023-03-31

- Improve bash ! syntax highlighting (GitHub issue)
- Fix bug where VIM keybindings weren't working in the file editor
- Upgraded R from 4.2.2 to 4.2.3
- Python package updates:
  - arviz 0.12.1 --> 0.15.1
  - astropy 4.3.1 --> 5.2.2
  - dopamine-rl 1.0.5 --> 4.0.6
  - gensim 3.6.0 --> 4.3.1
  - ipykernel 5.3.4 -> 5.5.6
  - ipython 7.9.0 -> 7.34.0
  - jax 0.4.4 -> 0.4.7
  - jaxlib 0.4.4 -> 0.4.7
  - jupyter_core 5.2.0 -> 5.3.0
  - keras 2.11.0 -> 2.12.0
  - lightgbm 2.2.3 -> 3.3.5
  - matplotlib 3.5.3 -> 3.7.1
  - nltk 3.7 -> 3.8.1
  - opencv-python 4.6.0.66 -> 4.7.0.72
  - plotly 5.5.0 -> 5.13.1
  - pymc 4.1.4 -> 5.1.2
  - seaborn 0.11.2 -> 0.12.2
  - spacy 3.4.4 -> 3.5.1
  - sympy 1.7.1 -> 1.11.1
  - tensorboard 2.11.2 -> 2.12.0
  - tensorflow 2.11.0 -> 2.12.0
  - tensorflow-estimator 2.11.0 -> 2.12.0
  - tensorflow-hub 0.12.0 -> 0.13.0
  - torch 1.13.1 -> 2.0.0
  - torchaudio 0.13.1 -> 2.0.1
  - torchtext 0.14.1 -> 0.15.1
  - torchvision 0.14.1 -> 0.15.1

2023-03-10

- Added the Colab editor shortcuts example notebook
- Fixed triggering of @-mention and email autocomplete for large comments (GitHub issue)
- Added View Resources to the Runtime menu
- Made file viewer images fit the view by default, resizing to original size on click
- When in VIM mode, enable copy as well as allowing propagation to monaco-vim to escape visual mode (GitHub issue)
- Upgraded CUDA 11.6.2 -> 11.8.0 and cuDNN 8.4.0.27 -> 8.7.0.84
- Upgraded Nvidia drivers 525.78.01 -> 530.30.02
- Upgraded Python 3.8.10 -> 3.9.16
- Python package updates:
  - beautifulsoup4 4.6.3 -> 4.9.3
  - bokeh 2.3.3 -> 2.4.3
  - debugpy 1.0.0 -> 1.6.6
  - Flask 1.1.4 -> 2.2.3
  - jax 0.3.25 -> 0.4.4

They do not agree. For the top 5, Ramada by Wyndham Cornwall is included in the mean ground truth top 5 but is not included in the mean vader top 5. Also, First Canada Inns is included in the mean ground truth, but not the median vader. Instead, Regency Inn & Suites is included in both the mean and median vader compound scores top 5.

This is due to the Ramada hotel having lots of reviews (582), most of which are good (4-5 ground truth), compared to Regency which has very few reviews (5) with a lower average true score. However, due to the small number of reviews, it's easily thrown off by outliers; one such outlier exists where the true score was 3 but the vader socre was very high (0.9155, same as the true scores of 4). The review sounded extremely positive and only mentioned good things.

For First Canada Inns, there are more reviews (126) so it's a bit less affected by such outliers. But in general, it seems to have a bit more neutral sentiment scores than Ramada hotel, which skews down the median vader score, but doesn't move the overall mean by as much due to the heavy positive skew still.

The individual placing within the top 5 changes, but that is expected due to differences in the distributions and differences between the ground truth score and the vader compound score.

```
df[np.logical_or(np.logical_or(df['hotelName'] == 'Regency Inn &amp; Suites', df['hote
```

```
hotelName
First Canada Inns             126
Ramada by Wyndham Cornwall    582
Regency Inn &amp; Suites        5
dtype: int64
```

```
df[np.logical_or(np.logical_or(df['hotelName'] == 'Regency Inn &amp; Suites', df['hote
```

```
hotelName                   ratingScore
First Canada Inns           1                9
                            2                8
                            3               17
                            4               30
                            5               62
Ramada by Wyndham Cornwall  1               29
                            2               36
                            3               86
                            4              210
                            5              221
Regency Inn &amp; Suites    1                1
                            3                1
                            4                3
dtype: int64
```

```
df[df['hotelName'] == 'Regency Inn &amp; Suites']
```

- jaxlib 0.3.25 -> 0.4.4
- Jinja2 2.11.3 -> 3.1.2
- matplotlib 3.2.2 -> 3.5.3
- nbconvert 5.6.1 -> 6.5.4
- pandas 1.3.5 -> 1.4.4
- pandas-datareader 0.9.0 -> 0.10.0
- pandas-profiling 1.4.1 -> 3.2.0
- Pillow 7.1.2 -> 8.4.0
- plotnine 0.8.0 -> 0.10.1
- scikit-image 0.18.3 -> 0.19.3
- scikit-learn 1.0.2 -> 1.2.2
- scipy 1.7.3 -> 1.10.1
- setuptools 57.4.0 -> 63.4.3
- sklearn-pandas 1.8.0 -> 2.2.0
- statsmodels 0.12.2 -> 0.13.5
- urllib3 1.24.3 -> 1.26.14
- Werkzeug 1.0.1 -> 2.2.3
- wrapt 1.14.1 -> 1.15.0
- xgboost 0.90 -> 1.7.4
- xlrd 1.2.0 -> 2.0.1

2023-02-17

- Show graphs of RAM and disk usage in notebook toolbar
- Copy cell links directly to the clipboard instead of showing a dialog when clicking on the link icon in the cell toolbar
- Updated the Colab Marketplace VM image
- Upgraded CUDA to 11.6.2 and cuDNN to 8.4.0.27
- Python package updates:
  - tensorflow 2.9.2 -> 2.11.0
  - tensorboard 2.9.1 -> 2.11.2
  - keras 2.9.0 -> 2.11.0
  - tensorflow-estimator 2.9.0 -> 2.11.0
  - tensorflow-probability 0.17.0 -> 0.19.0
  - tensorflow-gcs-config 2.9.0 -> 2.11.0
  - earthengine-api 0.1.339 -> 0.1.341
  - flatbuffers 1.12 -> 23.1.21
  - platformdirs 2.6.2 -> 3.0.0
  - pydata-google-auth 1.6.0 -> 1.7.0
  - python-utils 3.4.5 -> 3.5.2
  - tenacity 8.1.0 -> 8.2.1
  - tifffile 2023.1.23.1 -> 2023.2.3
  - notebook 5.7.16 -> 6.3.0
  - tornado 6.0.4 -> 6.2
  - aiohttp 3.8.3 -> 3.8.4
  - charset-normalizer 2.1.1 -> 3.0.1
  - fastai 2.7.0 -> 2.7.1
  - soundfile 0.11.0 -> 0.12.1
  - typing-extensions 4.4.0 -> 4.5.0
  - widgetsnbextension 3.6.1 -> 3.6.2
  - pydantic 1.10.4 -> 1.10.5
  - zipp 3.12.0 -> 3.13.0
  - numpy 1.21.6 -> 1.22.4
  - drivefs 66.0 -> 69.0
  - gdal 3.0.4 -> 3.3.2 GitHub issue
- Added libudunits2-dev for smoother R package installs GitHub issue

2023-02-03

- Improved tooltips for pandas series to show common statistics about the series object
- Made the forms dropdown behave like an autocomplete box when it allows input
- Updated the nvidia driver from 460.32.03 to 510.47.03
- Python package updates:
  - absl-py 1.3.0 -> 1.4.0
  - bleach 5.0.1 -> 6.0.0
  - cachetools 5.2.1 -> 5.3.0
  - cmdstanpy 1.0.8 -> 1.1.0
  - dnspython 2.2.1 -> 2.3.0
  - fsspec 2022.11.0 -> 2023.1.0
  - google-cloud-bigquery-storage 2.17.0 -> 2.18.1
  - holidays 0.18 -> 0.19
  - jupyter-core 5.1.3 -> 5.2.0
  - packaging 21.3 -> 23.0
  - prometheus-client 0.15.0 -> 0.16.0
  - pyct 0.4.8 -> 0.5.0
  - pydata-google-auth 1.5.0 -> 1.6.0
  - python-slugify 7.0.0 -> 8.0.0
  - sqlalchemy 1.4.46 -> 2.0.0
  - tensorflow-io-gcs-filesystem 0.29.0 -> 0.30.0

```
sns.violinplot(data=df[np.logical_or(df['hotelName'] == 'First Canada Inns', df['hotel
```

- tifffile 2022.10.10 -> 2023.1.23.1
- zipp 3.11.0 -> 3.12.0
- Pinned sqlalchemy to version 1.4.46

2023-01-12

- Added support for @-mention and email autocomplete in comments
- Improved errors when GitHub notebooks can't be loaded
- Increased color contrast for colors used for syntax highlighting in the code editor
- Added terminal access for custom GCE VM runtimes
- Upgraded Ubuntu from 18.04 LTS to 20.04 LTS (GitHub issue)
- Python package updates:
  - GDAL 2.2.2 -> 2.2.3.
  - NumPy from 1.21.5 to 1.21.6.
  - attrs 22.1.0 -> 22.2.0
  - chardet 3.0.4 -> 4.0.0
  - cloudpickle 1.6.0 -> 2.2.0
  - filelock 3.8.2 -> 3.9.0
  - google-api-core 2.8.2 -> 2.11.0
  - google-api-python-client 1.12.11 -> 2.70.0
  - google-auth-httplib2 0.0.3 -> 0.1.0
  - google-cloud-bigquery 3.3.5 -> 3.4.1
  - google-cloud-datastore 2.9.0 -> 2.11.0
  - google-cloud-firestore 2.7.2 -> 2.7.3
  - google-cloud-storage 2.5.0 -> 2.7.0
  - holidays 0.17.2 -> holidays 0.18
  - importlib-metadata 5.2.0 -> 6.0.0
  - networkx 2.8.8 -> 3.0
  - opencv-python-headless 4.6.0.66 -> 4.7.0.68
  - pip 21.1.3 -> 22.04
  - pip-tools 6.2.0 -> 6.6.2
  - prettytable 3.5.0 -> 3.6.0
  - requests 2.23.0 -> 2.25.1
  - termcolor 2.1.1 -> 2.2.0
  - torch 1.13.0 -> 1.13.1
  - torchaudio 0.13.0 -> 0.13.1
  - torchtext 0.14.0-> 0.14.1
  - torchvision 0.14.0 -> 0.14.1

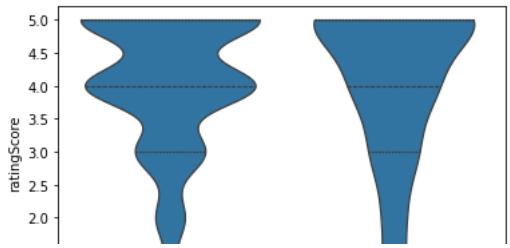
2022-12-06

- Made fallback runtime version available until mid-December (GitHub issue)
- Upgraded to Python 3.8 (GitHub issue)
- Python package updates:
  - jax from 0.3.23 to 0.3.25, jaxlib from 0.3.22 to 0.3.25
  - pyarrow from 6.0.1 to 9.0.0
  - torch from 1.12.1 to 1.13.0
  - torchaudio from 0.12.1 to 0.13.0
  - torchvision from 0.13.1 to 0.14.0
  - torchtext from 0.13.1 to 0.14.0
  - xlrd from 1.1.0 to 1.2.0
  - DriveFS from 62.0.1 to 66.0.3
- Made styling of markdown tables in outputs match markdown tables in text cells
- Improved formatting for empty interactive table rows
- Fixed syntax highlighting for variables with names that contain Python keywords (GitHub issue)

2022-11-11

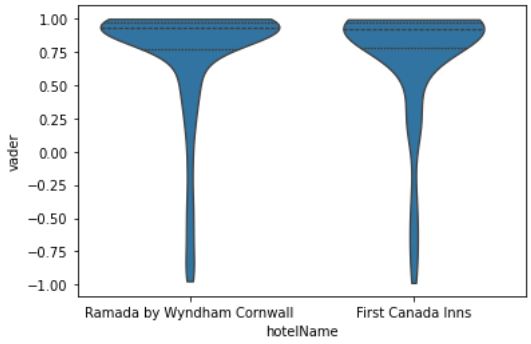
- Added more dark editor themes for Monaco (when in dark mode, "Editor colorization" appears as an option in the Editor tab of the Tools → Settings dialog)
- Fixed bug where collapsed forms were deleted on mobile GitHub issue
- Python package updates:
  - rpy2 from 3.4.0 to 3.5.5 (GitHub issue)
  - notebook from 5.5.0 to 5.7.16
  - tornado from 5.1.1 to 6.0.4
  - tensorflow_probability from 0.16.0 to 0.17.0
  - pandas-gbq from 0.13.3 to 0.17.9
  - protobuf from 3.17.3 to 3.19.6
  - google-api-core[grpc] from 1.31.5 to 2.8.2
  - google-cloud-bigquery from 1.21.0 to 3.3.5
  - google-cloud-core from 1.0.1 to 2.3.2
  - google-cloud-datastore from 1.8.0 to 2.9.0
  - google-cloud-firestore from 1.7.0 to 2.7.2
  - google-cloud-language from 1.2.0 to 2.6.1
  - google-cloud-storage from 1.18.0 to 2.5.0

```
<Axes: xlabel='hotelName', ylabel='ratingScore'>
```




```
sns.violinplot(data=df[np.logical_or(df['hotelName'] == 'First Canada Inns', df['hotel
```

```
<Axes: xlabel='hotelName', ylabel='vader'>
```



BOTTOM 5

```
bottom5 = pd.concat([bottom_5_true,bottom_5_vader_mean,bottom_5_vader_median], axis=1)
bottom5.columns = ['mean ground truth','mean vader compound','median vader compound']
bottom5
```

	mean ground truth	mean vader compound	median vader compound
10	Nites Inn Motel	Indigo Inn	Indigo Inn
11	Indigo Inn	Nites Inn Motel	Martin's Inn
12	Monte Carlo Motel	Howard Johnson by Wyndham Cornwall	Howard Johnson by Wyndham Cornwall
13	OYO First Canada Hotel Cornwall Hwy 401 ON	Monte Carlo Motel	Monte Carlo Motel

There is only one difference in the bottom 5 between the 3 ranking methods. The median vader compound doesn't have Nites Inn Motel, while it instead has Martin's Inn.

The reason for this is just due to the instability of the median for low sample sizes. Nites Inn Motel has only 7 reviews. 4 of them have very high vader scores (>0.79), while 3 of them have very negative scores (<-0.83). Given this, the median (~0.79) is a much larger number than the average, and makes it seem like the hotel is much better than the other 2 metrics.

Once again, the rankings within the top 5 also differ slightly, but that's expected due to differences in the distributions and the differences between the metrics of ground truth and vader sentiment.

```
df[np.logical_or(df['hotelName'] == 'Nites Inn Motel', df['hotelName'] == 'Martin&#39;
```

```
hotelName         ratingScore
Martin&#39;s Inn  1              4
                  2              2
                  3              2
                  4              2
                  5              2
Nites Inn Motel   1              3
                  3              2
                  4              2
dtype: int64
```

- google-cloud-translate from 1.5.0 to 3.8.4

2022-10-21

- Launched a single-click way to get from BigQuery to Colab to further explore query results (announcement)
- Launched Pro, Pro+, and Pay As You Go to 19 additional countries: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Denmark, Estonia, Finland, Greece, Hungary, Latvia, Lithuania, Norway, Portugal, Romania, Slovakia, Slovenia, and Sweden (tweet)
- Updated jax from 0.3.17 to 0.3.23, jaxlib from 0.3.15 to 0.3.22, TensorFlow from 2.8.2 to 2.9.2, CUDA from 11.1 to 11.2, and cuDNN from 8.0 to 8.1 (backend-info)
- Added a `readonly` option to `drive.mount`
- Fixed bug where Xarray was not working (GitHub issue)
- Modified Markdown parsing to ignore block quote symbol within MathJax (GitHub issue)

2022-09-30

- Launched Pay As You Go, allowing premium GPU access without requiring a subscription
- Added vim and tcllib to our runtime image
- Fixed bug where open files were closed on kernel disconnect (GitHub issue)
- Fixed bug where the play button/execution indicator was not clickable when scrolled into the cell output (GitHub issue)
- Updated the styling for form titles so that they avoid obscuring the code editor
- Created a GitHub repo, backend-info, with the latest apt-list.txt and pip-freeze.txt files for the Colab runtime (GitHub issue)
- Added `files.upload_file(filename)` to upload a file from the browser to the runtime with a specified filename

2022-09-16

- Upgraded pymc from 3.11.0 to 4.1.4, jax from 0.3.14 to 0.3.17, jaxlib from 0.3.14 to 0.3.15, fsspec from 2022.8.1 to 2022.8.2
- Modified our save flow to avoid persisting Drive filenames as titles in notebook JSON
- Updated our Terms of Service
- Modified the `Jump to Cell` command to locate the cursor at the end of the command palette input (`Jump to cell` in Tools → Command palette in a notebook with section headings)
- Updated the styling of the Drive notebook comment UI
- Added support for terminating your runtime from code: `python from google.colab import runtime runtime.unassign()`
- Added regex filter support to the Recent notebooks dialog
- Inline google.colab.files.upload JS to fix `files.upload()` not working (GitHub issue)

2022-08-26

- Upgraded PyYAML from 3.13 to 6.0 (GitHub issue), drivefs from 61.0.3 to 62.0.1
- Upgraded TensorFlow from 2.8.2 to 2.9.1 and ipywidgets from 7.7.1 to 8.0.1 but rolled both back due to a number of user reports (GitHub issue, GitHub issue)
- Stop persisting inferred titles in notebook JSON (GitHub issue)
- Fix bug in background execution which affected some Pro+ users (GitHub issue)
- Fix bug where `Download as .py` incorrectly handled text cells ending in a double quote
- Fix bug for Pro and Pro+ users where we weren't honoring the preference (Tools → Settings) to use a temporary scratch notebook as the default landing page
- Provide undo/redo for scratch cells
- When writing ipynb files, serialize empty multiline strings as `[]` for better consistency with JupyterLab

2022-08-11

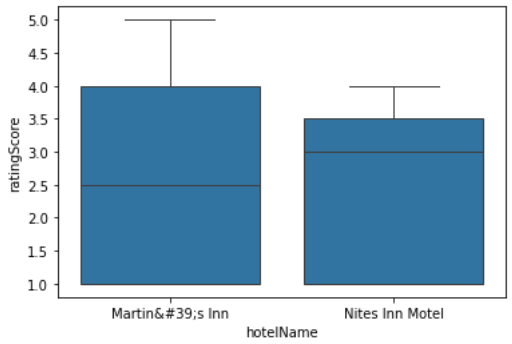
- Upgraded ipython from 5.5.0 to 7.9.0, fbprophet 0.7 to prophet 1.1, tensorflow-datasets from 4.0.1 to 4.6.0, drivefs from 60.0.2 to 61.0.3, pytorch from 1.12.0 to 1.12.1, numba from 0.51 to 0.56, and lxml from 4.2.0 to 4.9.1

```
df[df['hotelName'] == 'Nites Inn Motel'].sort_values('vader')['vader']
```

```
2271   -0.9513
2273   -0.8779
2275   -0.8342
2270    0.7960
2269    0.8272
2274    0.8422
2272    0.9359
Name: vader, dtype: float64
```

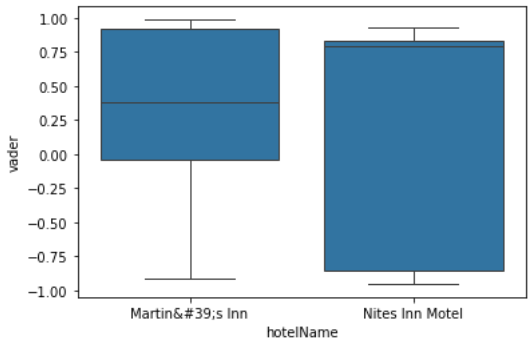
```
#fig, ax = plt.subplots(figsize=[20,5])
#plt.xticks(rotation=90)
sns.boxplot(data=df[np.logical_or(df['hotelName'] == 'Nites Inn Motel', df['hotelName'
```

```
<Axes: xlabel='hotelName', ylabel='ratingScore'>
```




```
sns.boxplot(data=df[np.logical_or(df['hotelName'] == 'Nites Inn Motel', df['hotelName'
```

```
<Axes: xlabel='hotelName', ylabel='vader'>
```



Q2

- Loosened our `requests` version requirement (GitHub issue)
- Removed support for TensorFlow 1
- Added Help → Report Drive abuse for Drive notebooks
- Fixed indentation for Python lines ending in [
- Modified styling of tables in Markdown to left-align them rather than centering them
- Fixed special character replacement when copying interactive tables as Markdown
- Fixed ansi 8-bit color parsing (GitHub issue)
- Configured logging to preempt transitive imports and other loading from implicitly configuring the root logger
- Modified forms to use a value of `None` instead of causing a parse error when clearing raw and numeric-typed form fields

2022-07-22

- Update scipy from 1.4.1 to 1.7.3, drivefs from 59.0.3 to 60.0.2, pytorch from 1.11 to 1.12, jax & jaxlib from 0.3.8 to 0.3.14, opencv-python from 4.1.2.30 to 4.6.0.66, spaCy from 3.3.1 to 3.4.0, and dlib from 19.18.0 to 19.24.0
- Fix `Open in tab` doc link which was rendering incorrectly (GitHub issue)
- Add a preference for the default tab orientation to the Site section of the settings menu under Tools → Settings
- Show a warning for `USE_AUTH_EPHEM` usage when running authenticate_user on a TPU runtime (code)

2022-07-01

- Add a preference for code font to the settings menu under Tools → Settings
- Update drivefs from 58.0.3 to 59.0.3 and spacy from 2.2.4 to 3.3.1
- Allow display_data and execute_result text outputs to wrap, matching behavior of JupyterLab (does not affect stream outputs/print statements).
- Improve LSP handling of some magics, esp. %%writefile (GitHub issue).
- Add a FAQ entry about the mount Drive button behavior and include link buttons for each FAQ entry.
- Fix bug where the notebook was sometimes hidden behind other tabs on load when in single pane view.
- Fix issue with inconsistent scrolling when an editor is in multi-select mode.
- Fix bug where clicking on a link in a form would navigate away from the notebook
- Show a confirmation dialog before performing Replace all from the Find and replace pane.

2022-06-10

- Update drivefs from 57.0.5 to 58.0.3 and tensorflow from 2.8.0 to 2.8.2
- Support more than 100 repos in the GitHub repo selector shown in the open dialog and the clone to GitHub dialog
- Show full notebook names on hover in the open dialog
- Improve the color contrast for links, buttons, and the `ipywidgets.Accordion` widget in dark mode

2022-05-20

- Support URL params for linking to some common pref settings: force_theme=dark, force_corgi_mode=1, force_font_size=14. Params forced by URL are not persisted unless saved using Tools → Settings.
- Add a class `markdown-google-sans` to allow Markdown to render in Google Sans
- Update monaco-vim from 0.1.19 to 0.3.4
- Update drivefs from 55.0.3 to 57.0.5, jax from 0.3.4 to 0.3.8, and jaxlib from 0.3.2 to 0.3.7

2022-04-29

- Added 🐞 mode (under Miscellaneous in Tools → Settings)
- Added "Disconnect and delete runtime" option to the menu next to the Connect button
- Improved rendering of filter options in an interactive table
- Added git-lfs to the base image

```
def get_stop_words():
    stop = set(stopwords.words('english'))
    #Add possible Stop Words for Hotel Reviews
    stop.add('hotel')
    stop.add('room')
    stop.add('rooms')
    stop.add('stay')
    stop.add('staff')
    return stop

def getTopKWords(df, kwords):

    stop = get_stop_words()
    counter = Counter()

    reviews = df['reviewColumn'].values

    for review in reviews:
            counter.update([word.lower()
                            for word
                            in re.findall(r'\w+', review)
                            if word.lower() not in stop and len(word) > 2])
    topk = counter.most_common(kwords)
    return topk

# Note: You may want to use an NLTK tokenizer instead of a regular expression in the f
def dataFrameTransformation(df, topk):
    reviews = df['reviewColumn'].values

    #Find out if a particular review has the word from topk list
    freqReview = []
    for i in range(len(reviews)):
        tempCounter = Counter([word.lower() for word in re.findall(r'\w+',reviews[i])]
        topkinReview = [1 if tempCounter[word] > 0 else 0 for (word,wordCount) in topk
        freqReview.append(topkinReview)


    #Prepare freqReviewDf
    freqReviewDf = pd.DataFrame(freqReview)
    dfName = []
    for c in topk:
        dfName.append(c[0])
    freqReviewDf.columns = dfName
    #finalreviewDf = df.join(freqReviewDf)
    finaldf = df[['hotelName','ratingScore','groundTruth','reviewColumn','vader']].joi
    return finaldf


reviewDF
```

- Updated torch from 1.10.0 to 1.11.0, jupyter-core from 4.9.2 to 4.10.0, and cmake from 3.12.0 to 3.22.3
- Added more details to our FAQ about unsupported uses (using proxies, downloading torrents, etc.)
- Fixed issue with apt-get dependencies

2022-04-15

- Add an option in the file browser to show hidden files.
- Upgrade gdown from 4.2.0 to 4.4.0, google-api-core[grpc] from 1.26.0 to 1.31.5, and pytz from 2018.4 to 2022.1

2022-03-25

- Launched Pro/Pro+ to 12 additional countries: Australia, Bangladesh, Colombia, Hong Kong, Indonesia, Mexico, New Zealand, Pakistan, Philippines, Singapore, Taiwan, and Vietnam
- Added `google.colab.auth.authenticate_service_account(` to support using Service Account keys
- Update jax from 0.3.1 to 0.3.4 & jaxlib from 0.3.0 to 0.3.2
- Fixed an issue with Twitter previews of notebooks shared as Github Gists

2022-03-10

- Launched Pro/Pro+ to 10 new countries: Ireland, Israel, Italy, Morocco, the Netherlands, Poland, Spain, Switzerland, Turkey, and the United Arab Emirates
- Launched support for scheduling notebooks for Pro+ users
- Fixed bug in interactive datatables where filtering by number did not work
- Finished removing the python2 kernelspec

2022-02-25

- Made various accessibility improvements to the header
- Fix bug with forms run:auto where a form field change would trigger multiple runs
- Minor updates to the bigquery example notebook and snippet
- Include background execution setting in the sessions dialog for Pro+ users
- Update tensorflow-probability from 0.15 to 0.16
- Update jax from 0.2.25 to 0.3.1 & jaxlib from 0.1.71 to 0.3.0

2022-02-11

- Improve keyboard navigation for the open dialog
- Fix issue where nvidia-smi stopped reporting resource utilization for some users who were modifying the version of nvidia used
- Update tensorflow from 2.7 to 2.8, keras from 2.7 to 2.8, numpy from 1.19.5 to 1.21.5, tables from 3.4.4 to 3.7.0

2022-02-04

- Improve UX for opening content alongside your notebook, such as files opened from the file browser. This includes a multi-pane view and drag-drop support
- Better Twitter previews when sharing example Colab notebooks and notebooks opened from GitHub Gists
- Update pandas from 1.1.5 to 1.3.5
- Update openpyxl from 2.5.9 to 3.0.0 and pyarrow from 3.0.0 to 6.0.0
- Link to the release notes from the Help menu

2022-01-28

- Add a copy button to data tables
- Python LSP support for better completions and code diagnostics. This can be configured in the Editor Settings (Tools → Settings)
- Update gspread examples in our documentation
- Update gdown from 3.6 to 4.2

2022-01-21

- New documentation for the `google.colab` package
- Show GPU RAM in the resource usage tab
- Improved security for mounting Google Drive which disallows mounting Drive from accounts other than the one currently executing the notebook

	reviewCol	vader
0	"Should have known better . upon checking in was told that service at Christmas \u002F Boxing Day would be minimal. DID not know that meant NON existent. We never got room clean up \u002F towel refresh or any service at all for the 1 week we were their. December 20 to December 27th. On day 2 the husband told us his wife would be around to change the bedding and towels on Day 3 .. that never happened. When I called the front desk I was told I had to come get my own clean	-0.1653

```
# setting default parameters of WordCloud object
wordcloud_args = dict(
    width = 800,
    height = 800,
    background_color ='white',
    min_font_size = 10
    )

# fucntion to plot word cloud
def plotWordCloud(dictionary, **kwargs):
    wordcloud = WordCloud(**kwargs)
    wordcloud.generate_from_frequencies(dict(dictionary))
    plt.figure()
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.show()
```

```
topk_pos = getTopKWords(df[df['groundTruth']=='positive'], 50)
topk_pos
```

```
[('breakfast', 1308),
 ('clean', 1170),
 ('great', 1024),
 ('good', 841),
 ('friendly', 767),
 ('comfortable', 759),
 ('nice', 752),
 ('cornwall', 684),
 ('would', 666),
 ('stayed', 630),
 ('well', 551),
 ('best', 508),
 ('restaurant', 472),
 ('bed', 471),
 ('place', 466),
 ('one', 462),
 ('pool', 456),
 ('excellent', 434),
 ('area', 431),
 ('night', 431),
 ('service', 412),
 ('helpful', 381),
 ('time', 353),
 ('food', 337),
 ('recommend', 337),
 ('desk', 326),
 ('also', 298),
 ('front', 292),
 ('beds', 289),
 ('back', 288),
 ('like', 282),
 ('quiet', 274),
 ('inn', 274),
 ('western', 274),
 ('nthe', 273),
 ('location', 271),
 ('check', 270),
 ('way', 266),
 ('hot', 259),
 ('fireplace', 256),
 ('always', 255),
 ('large', 243),
 ('buffet', 242),
 ('get', 239),
 ('included', 232),
 ('definitely', 232),
 ('could', 232),
 ('really', 226),
 ('next', 223),
 ('home', 222)]
```

2022-01-14

- Add a preference (Tools → Settings) to use a temporary scratch notebook as the default landing page
- Fix bug where `/` and `:` weren't working in VIM mode
- Update gspread from 3.0 to 3.4
- Update the Colab Marketplace VM image

```
plotWordCloud(topk_pos, **wordcloud_args)
```

hotels on the strip,

```
topk_neg = getTopKWords(df[df['groundTruth']=='negative'], 50)
topk_neg
```

```
[('breakfast', 484),
 ('clean', 389),
 ('would', 369),
 ('one', 355),
 ('night', 341),
 ('good', 285),
 ('desk', 284),
 ('bed', 280),
 ('front', 261),
 ('stayed', 234),
 ('place', 232),
 ('could', 228),
 ('like', 223),
 ('get', 212),
 ('pool', 206),
 ('nthe', 198),
 ('back', 193),
 ('nice', 191),
 ('bathroom', 189),
 ('door', 189),
 ('time', 177),
 ('friendly', 172),
 ('floor', 171),
 ('cornwall', 166),
 ('motel', 153),
 ('even', 152),
 ('old', 151),
 ('area', 150),
 ('coffee', 149),
 ('check', 148),
 ('price', 145),
 ('well', 144),
 ('comfortable', 141),
 ('told', 139),
 ('beds', 139),
 ('small', 137),
 ('went', 136),
 ('booked', 133),
 ('great', 131),
 ('first', 129),
 ('never', 126),
 ('dirty', 123),
 ('got', 121),
 ('also', 121),
 ('said', 117),
 ('next', 117),
 ('asked', 115),
 ('way', 112),
 ('morning', 110),
 ('smell', 105)]
```

expecting much

```
plotWordCloud(topk_neg, **wordcloud_args)
```

model of that

```
combined_words = set([i[0] for i in topk_pos] + [i[0] for i in topk_neg])
combined_count = Counter(combined_words)
for i in topk_pos+topk_neg:
    if i[0] in combined_words:
        combined_count[i[0]] += i[1]
```


\nThe motel is very

```
combined_count.most_common(10)
```

```
[('breakfast', 1793),
 ('clean', 1560),
 ('great', 1156),
 ('good', 1127),
 ('would', 1036),
 ('nice', 944),
 ('friendly', 940),
 ('comfortable', 901),
 ('stayed', 865),
 ('cornwall', 851)]
```

```
', '.join([i[0] for i in combined_count.most_common(10)])
```

```
'breakfast, clean, great, good, would, nice, friendly, comfortable, stayed,
cornwall'
```

Cornwall appears in both lists of top-50 words, which is the location of the hotels.

Many words appear in both top-50 lists. The top 10 most common shared words are breakfast, clean, great, good, would, nice, friendly, comfortable, stayed, and cornwall.

Of these, the words "clean" and "good" are similarly large in both wordclouds. This is surprising to me, since both words have positive connotations. However, it makes sense that they might be used with a negative modifier, which explains the appearance in both lists.

```
grammar1 = r"""
    NBAR:
        {<NN.*|JJ>*<NN.*>}
    NP:
        {<NBAR><IN><NBAR>}
        {<NBAR>}
"""
```

```
# to make the results more useable, we clean up the tree results shown above.
lemmatizer = nltk.WordNetLemmatizer()
stemmer = nltk.stem.porter.PorterStemmer()
stopword_list = get_stop_words()

# generator, create item one a time
def get_terms(tree, remove_stopwords=True):
    for leaf in leaves(tree):

        term = []

        for w, t in leaf:
            if remove_stopwords:
                if acceptable_word(w):
                    term.append(normalise(w))

            else:
                term.append(normalise(w))

        # Phrase only
        if len(term) > 1:
            yield term

# generator, generate leaves one by one
def leaves(tree):
    """Finds NP (nounphrase) leaf nodes of a chunk tree."""
    for subtree in tree.subtrees(filter = lambda t: t.label()=='NP' or t.label()=='JJ'
        yield subtree.leaves()

# stemming, lematizing, lower case...
def normalise(word,lemmatizer=lemmatizer, stemmer=stemmer):
    """Normalises words to lowercase and stems and lemmatizes it."""
    word = word.lower()
    word = stemmer.stem(word)
    word = lemmatizer.lemmatize(word)
    return word

# stop-words and length control
def acceptable_word(word, stopword_list=stopword_list):
    """Checks conditions for acceptable word: length, stopword."""
    accepted = bool(2 <= len(word) <= 40
        and word.lower() not in stopword_list)
    return accepted

# Flatten phrase lists to get tokens for analysis
def flatten_phrase_lists(npTokenList):
    finalList =[]
    for phrase in npTokenList:
        token = ''
        for word in phrase:
            token += word + ' '
        if len(token) > 0: # don't return empty tokens
            finalList.append(token.rstrip())
    return finalList
```

```
# getting top noun phrases
tagger = PerceptronTagger()
pos_tag = tagger.tag
def getTopKNP(df, kNPs, grammar):
    chunker = nltk.RegexpParser(grammar)
    counter = Counter()
    reviews = df['reviewColumn'].values

    for review in reviews:
            counter.update(flatten_phrase_lists([word
                            for word
                            in get_terms(chunker.parse(pos_tag(re.findall(r'\w+', revi
                            ]))
    topk = counter.most_common(kNPs)
    return topk


def NPdataFrameTransformation(df, topk, grammar):
    chunker = nltk.RegexpParser(grammar)
    reviews = reviewDF['reviewCol'].values

    #Find out if a particular review has the word from topk list
    freqReview = []
    for i in range(len(reviews)):
        tempCounter = Counter(flatten_phrase_lists([word
                                    for word
                                    in get_terms(chunker.parse(pos_tag(re.findall(r
        topkinReview = [1 if tempCounter[word] > 0 else 0 for (word,wordCount) in topk
        freqReview.append(topkinReview)


    #Prepare freqReviewDf
    freqReviewDf = pd.DataFrame(freqReview)
    dfName = []
    for c in topk:
        dfName.append(c[0])
    freqReviewDf.columns = dfName
    #finalreviewDf = df.join(freqReviewDf)
    finaldf = df[['hotelName','ratingScore','groundTruth','reviewColumn','vader']].joi
    return finaldf


topk_pos = getTopKNP(df[df['groundTruth'] == 'positive'], 50, grammar1)
topk_pos
```

```
 [('front desk', 140),
  ('best western', 73),
  ('breakfast buffet', 43),
  ('continent breakfast', 42),
  ('hot tub', 42),
  ('comfort bed', 40),
  ('great place', 39),
  ('second floor', 36),
  ('next morn', 35),
  ('free breakfast', 33),
  ('first time', 29),
  ('ramada cornwal', 29),
  ('park lot', 28),
  ('first floor', 27),
  ('queen bed', 26),
  ('ground floor', 25),
  ('next time', 25),
  ('comfort inn', 24),
  ('outdoor pool', 24),
  ('complimentari breakfast', 23),
  ('sport bar', 23),
  ('long day', 23),
  ('mani time', 21),
  ('buffet breakfast', 21),
  ('great locat', 19),
  ('nice touch', 18),
  ('good valu', 18),
  ('breakfast area', 18),
  ('hot breakfast', 18),
  ('long drive', 18),
  ('indoor pool', 18),
  ('cornwal area', 17),
  ('way home', 17),
  ('king bed', 17),
  ('tim horton', 17),
```

```
 ('coffe maker', 16),
 ('next day', 16),
 ('easi access', 16),
 ('quebec citi', 15),
 ('second time', 15),
 ('great valu', 14),
 ('nice place', 14),
 ('next year', 14),
 ('main floor', 14),
 ('great servic', 14),
 ('good size', 14),
 ('pleasant surpris', 14),
 ('king suit', 14),
 ('clean bed', 14),
 ('next door', 13)]
```

```
plotWordCloud(topk_pos, **wordcloud_args)
```

The phrase "long drive" is unexpected to appear in the top 50 for positive. However, it may make sense if it's used wuth a negative, such as "not a long drive", which is a common phrase.

```
topk_neg = getTopKNP(df[df['groundTruth'] == 'negative'], 50, grammar1)
topk_neg
```

```
[('front desk', 139),
 ('comfort inn', 40),
 ('second floor', 32),
 ('park lot', 31),
 ('continent breakfast', 22),
 ('next morn', 21),
 ('hot tub', 19),
 ('queen bed', 18),
 ('ground floor', 18),
 ('tim horton', 16),
 ('coffe maker', 16),
 ('air condition', 15),
 ('first floor', 15),
 ('non smoke', 14),
 ('next day', 14),
 ('breakfast area', 12),
 ('best western', 12),
 ('howard johnson', 12),
 ('next time', 11),
 ('desk clerk', 11),
 ('free breakfast', 10),
 ('last time', 10),
 ('credit card', 10),
 ('wi fi', 9),
 ('good locat', 9),
 ('patio door', 9),
 ('waffl maker', 9),
 ('main floor', 9),
 ('pool area', 9),
 ('first night', 9),
 ('good thing', 9),
 ('hot breakfast', 9),
 ('east coast', 8),
 ('front desk clerk', 8),
 ('long day', 8),
 ('indoor pool', 8),
 ('complimentari breakfast', 8),
 ('ice machin', 8),
 ('hot water', 7),
 ('long time', 7),
 ('doubl bed', 7),
 ('cigarett smoke', 7),
```

```
      ('new owner', 6),
      ('air condit', 6),
      ('last night', 6),
      ('custom servic', 6),
      ('hair dryer', 6),
      ('ac unit', 6),
      ('hour drive', 6),
      ('nthe breakfast', 6)]
```

```
plotWordCloud(topk_neg, **wordcloud_args)
```


```
combined_words = set([i[0] for i in topk_pos] + [i[0] for i in topk_neg])
combined_count = Counter(combined_words)
for i in topk_pos+topk_neg:
    if i[0] in combined_words:
        combined_count[i[0]] += i[1]
', '.join([i[0] for i in combined_count.most_common(5)])
```

```
     'front desk, best western, second floor, comfort inn, continent breakfast'
```

Many NPs are included in both the top 50 positive and negative reviews. Some common words which are large in both word clouds are 'front desk, second floor, comfort inn, continent breakfast'

c)

```
grammar2 = r"""
    NBAR:
        {<NN.*|JJ>*<VB.*>*<JJ.*>}
    NP:
        {<NBAR><IN><NBAR>}
        {<NBAR>}
"""
```

```
topk_pos = getTopKNP(df[df['groundTruth'] == 'positive'], 50, grammar2)
topk_pos
```

```
     [('bed comfort', 71),
      ('best western', 39),
      ('make sure', 22),
      ('food excel', 20),
      ('year old', 19),
      ('clean comfort', 18),
      ('breakfast good', 17),
      ('breakfast delici', 17),
      ('breakfast great', 16),
      ('servic excel', 14),
      ('bit date', 13),
      ('littl date', 13),
      ('everyth clean', 11),
      ('breakfast excel', 11),
      ('flat screen', 10),
      ('clean friendli', 10),
      ('bed comfi', 10),
      ('food good', 10),
      ('servic great', 10),
      ('food great', 10),
      ('feel welcom', 9),
      ('pet friendli', 9),
      ('bathroom clean', 8),
      ('food delici', 8),
      ('locat perfect', 8),
```

```
('bed super', 8),
('felt safe', 8),
('nice clean', 8),
('clean quiet', 7),
('restaur excel', 7),
('locat good', 7),
('bit small', 6),
('price right', 6),
('front desk', 6),
('friendli help', 6),
('comfort clean', 6),
('bed great', 6),
('locat conveni', 5),
('arriv late', 5),
('place clean', 5),
('park right', 5),
('nice comfort', 5),
('includ hot', 5),
('say enough', 5),
('quiet comfort', 5),
('clean spaciou', 5),
('restaur great', 5),
('littl small', 5),
('hot tub great', 5),
('breakfast friendli', 4)]
```

```
plotWordCloud(topk_pos, **wordcloud_args)
```


```
topk_neg = getTopKNP(df[df['groundTruth'] == 'negative'], 50, grammar2)
topk_neg
```

```
[('bed comfort', 35),
 ('bathroom clean', 9),
 ('noth special', 8),
 ('breakfast terribl', 7),
 ('best western', 7),
 ('make sure', 6),
 ('bed uncomfort', 6),
 ('bed clean', 5),
 ('year old', 5),
 ('bed good', 5),
 ('look clean', 4),
 ('feel safe', 4),
 ('look like', 4),
 ('pet friendli', 4),
 ('pool great', 4),
 ('pool clean', 4),
 ('front desk', 4),
 ('pool nice', 4),
 ('breakfast good', 4),
 ('bit date', 4),
 ('get rid', 4),
 ('get better', 4),
 ('get readi', 4),
 ('place old', 3),
 ('sheet clean', 3),
 ('anyth special', 3),
 ('nthe clean', 3),
 ('smoke free', 3),
 ('breakfast fine', 3),
 ('clean comfort', 3),
 ('bed old', 3),
 ('servic good', 3),
 ('night nthe', 3),
 ('carpet filthi', 3),
 ('stifl hot', 2),
 ('month old', 2),
 ('wifi free', 2),
```

```
('bed u002f', 2),
('clean good', 2),
('queen bed clean', 2),
('clean old', 2),
('paper thin', 2),
('think next', 2),
('expect better', 2),
('front desk ladi', 2),
('front lobbi clean', 2),
('comfort good', 2),
('tea nthe', 2),
('price excel', 2),
('ankl deep', 2)]
```

```
plotWordCloud(topk_neg, **wordcloud_args)
```

The top 50 for both grammars include phrases about the cleanliness of the room, the comfort of the bed, the friendliness of the staf, and the breakfast. However, the new grammar includes more phrases expressing emotions or feelings towards topics, such as "breakfast terribl" instead of just mentioning the topic like "continent breakfast" which was frequent with the first grammar. The positive case also has many more phrases which express the quality of the services instead of just the topics. For example, the new grammar says "breakfast delici" instead of just "continent breakfast" or "breakfast buffet".

I think the second pattern will be better since it has more detailed phrases that more often contain words relating to the sentiment of the phrase. But, these phrases are more specific and therefore might be harder to generalize to future reviews outside the training set.

Q3

```
# get Top K mutual information terms from the dataframe
def getMI(topk, df, label_column='groundTruth'):
    miScore = []
    for word in topk:
        miScore.append([word[0]]+[metrics.mutual_info_score(df[label_column], df[word[
    miScoredf = pd.DataFrame(miScore).sort_values(1,ascending=0)
    miScoredf.columns = ['Word','MI Score']
    return miScoredf
```

```
topk = getTopKWords(df, 2000)
finaldf = dataFrameTransformation(df, topk)
miScoredf = getMI(topk, finaldf)
miScoredf[:50]['Word'].values
```

```
array(['dirty', 'great', 'excellent', 'told', 'smell', 'smoke',
       'fireplace', 'comfortable', 'said', 'restaurant', 'old',
       'disgusting', 'wonderful', 'carpet', 'asked', 'worst', 'musty',
       'smelled', 'refund', 'left', 'sheets', 'rude', 'never', 'night',
       'filthy', 'paid', 'broken', 'terrible', 'cleaned', 'smoking',
       'cigarette', 'nothing', 'carpets', 'buffet', 'could', 'friendly',
       'floor', 'loud', 'recommend', 'stained', 'looked', 'bathroom',
       'enjoyed', 'definitely', 'tired', 'always', 'worn', 'cheap', 'non',
       'someone'], dtype=object)
```

```
plotWordCloud(zip(miScoredf[:50]['Word'].values, miScoredf[:50]['MI Score'].values), *
```

The highest MI value words are "dirty", "great", and "excellent", which makes sense since a dirty room is very bad, while saying something is "great" or "excellent" is a strong positive sentiment.

```
topkNP = getTopKNP(df, 1000, grammar = grammar2)
finaldfNP = NPdataFrameTransformation(df, topkNP, grammar2)
miScoredfNP = getMI(topkNP, finaldfNP)
miScoredfNP[:50]['Word'].values
```

```
array(['bed uncomfort', 'noth special', 'food excel', 'bed clean',
       'breakfast terribl', 'breakfast delici', 'look clean', 'pool nice',
       'look like', 'get better', 'bed good', 'bed old', 'carpet filthi',
       'place old', 'everyth clean', 'food great', 'servic great',
       'flat screen', 'get rid', 'locat perfect', 'food delici',
       'bit stiff', 'guess ok', 'breakfast horribl', 'way overpr',
       'call free', 'thing clean', 'breakfast okay', 'paper thin',
       'stifl hot', 'ankl deep', 'bathroom light', 'front desk rude',
       'floor big', 'stair old', 'need major', 'expect low', 'door open',
       'eat nearbi', 'comfort flat screen', 'wifi free', 'price excel',
       'tea nthe', 'front lobbi clean', 'front desk ladi',
       'expect better', 'think next', 'decent littl', 'queen bed clean',
       'noth wrong'], dtype=object)
```

```
plotWordCloud(zip(miScoredfNP[:50]['Word'].values, miScoredfNP[:50]['MI Score'].values
```

Many of the top noun phrases in terms of MI were discussing the state of the bed: "bed uncomfort", "bed clean", "bed good", "bed old", etc. So, the state of the bed appears to be most important to customers, and hotels should ensure the bed is comfy and clean.

Some other topics of interest are breakfast ("breakfast terribl" and "breakfast delici"), cleanliness of other things ('carpet filthi', 'everyth clean', 'thing clean'), and services/ammenities ('pool nice', 'servic great', 'front desk rude'), so improving the breakfast options, ensuring everything is clean, and improving or including extra services/amenities seems to be the next best options for improving the hotel.

Q4

```
# Compute PMI for all terms and all possible labels
def pmiForAllCal(df, topk_word, gt_sentiment, label_column='groundTruth'):
    #Try calculate all the pmi for top k and store them into one pmidf dataframe

    index = [x[0] for x in topk_word]
    pmiDf = pd.DataFrame(index=index, columns=['pmi'])

    for (word, count) in tqdm(topk_word):
        pmiDf.at[word, 'pmi'] = pmiCalc(df,word,gt_sentiment,label_column)

    return pmiDf


def pmiCalc(df, word, gt_sentiment, label_column='groundTruth'):

    N = df.shape[0]

    px = sum(df[label_column]==gt_sentiment)
    py = sum(df[word]==True)
    pxy = len(df[(df[label_column]==gt_sentiment) & (df[word]==True)])

    if pxy==0:#Log 0 cannot happen
        pmi = math.log((pxy+0.0001)*N/(px*py))
    else:
        pmi = math.log(pxy*N/(px*py))
    return pmi


pmiposdf = pmiForAllCal(finaldf,topk,'positive').sort_values('pmi',ascending=0)

     100%|██████████| 2000/2000 [00:03<00:00, 651.62it/s]


pmiposdf[:50]
```

	pmi
genuine	0.325662
andrew	0.325662
restful	0.325662
cycling	0.325662
heritage	0.325662
meetings	0.325662
relaxed	0.325662
cute	0.325662
dine	0.325662
attended	0.325662
delighted	0.325662
westerns	0.325662
provides	0.325662
cookies	0.325662
favourite	0.325662
scottie	0.325662
jim	0.325662
bikes	0.325662
gift	0.325662
beautifully	0.325662
hosts	0.325662
delight	0.325662
gina	0.325662
music	0.325662

```
plotWordCloud(zip(pmiposdf[:50].index, pmiposdf[:50]['pmi'].values), **wordcloud_args)
```

centrally 0.325662

Andrew appears as a high PMI word: this is likely due to a location specific staff member called Andrew, and all reviews that mention him are positive. We also see "Jim" and "Gina", likely more staff members.

Genuine has a high PMI: this suggests that a customer believing the location or staff are genuine is valued very positively, as all reviews that mention it are positive. It is interesting since it doesn't show up in other lists, likely due to the low occurance of the word.

We also see "Halifax" show up in the top 50. This is very interesting: it just so happens that several of the hotels have reviews from people travelling to/from Halifax. Cornwall is a common city to pass through on the route between Halifax and Toronto, so this makes sense. All people mentioning Halifax have positive reviews, and there are a lot of possible explanations for this. It could just be that positive reviews tend to mention the locations more since they tend to be more conversational, light-hearted or focused on the enjoyment of the trip. It may also be that the long trip to/from Halifax improves perception of the hotel. Finally,

it may be that people travelling to/from Halifax tend to leave more positive reviews. Without more info, it's impossible to tell which explanation is correct or how much of a role it plays.

unique 0.325662

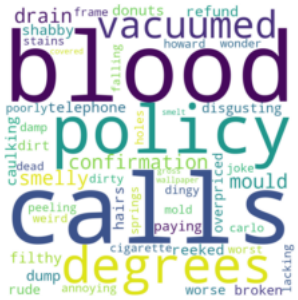
```
pminegdf = pmiForAllCal(finaldf,topk,'negative').sort_values('pmi',ascending=0)
```

```
     100%|██████████| 2000/2000 [00:02<00:00, 770.97it/s]
```

```
pminegdf[:50]
```

	pmi
calls	1.280311
blood	1.280311
policy	1.280311
degrees	1.280311
vacuumed	1.280311
confirmation	1.280311
drain	1.280311
mould	1.280311
smelly	1.280311
telephone	1.280311

```
plotWordCloud(zip(pminegdf[:50].index, pminegdf[:50]['pmi'].values), **wordcloud_args)
```



dirt	1.229018

Once again we see mention of staff, such as "Howard" and "Carlo", this time in a negative light

It's interesting to see policy have a high PMI for negative reviews, in general use it sounds pretty neutral but this shows that people mostly talk about policy when they dislike it.

Confirmation is much like policy; it seems to be mostly brought up when there's an issue, despite not sounding like a negative word in other context.

```
pmiposdf = pmiForAllCal(finaldfNP,topkNP,'positive').sort_values('pmi',ascending=0)
```

```
100%|██████████| 1000/1000 [00:00<00:00, 1069.79it/s]
```

peeling	1.215772

```
pmiposdf[:50]
```

	pmi
bathroom compact	0.325662
call next	0.325662
everyth wonder	0.325662
jacuzzi nice	0.325662
employe great	0.325662
rate good	0.325662
midnight music loud	0.325662
bit worn	0.325662
locat fabul	0.325662
breakfast buffet great	0.325662
call front	0.325662
tip top	0.325662
friendli clean	0.325662
bed huge	0.325662
clean help	0.325662
desk courteou	0.325662
great free	0.325662
front desk excel	0.325662
canopi bed clean	0.325662
jacuzzi larg	0.325662
help courteou	0.325662
decor clean	0.325662
pool tabl	0.325662
seem happi	0.325662
felt comfort	0.325662
parti next	0.325662
clean close	0.325662
great comfort	0.325662
shower excel	0.325662
server excel	0.325662
morn overal	0.325662
serv good	0.325662
food fresh	0.325662
breakfast buffet outstand	0.325662

```
plotWordCloud(zip(pmiposdf[:50].index, pmiposdf[:50]['pmi'].values), **wordcloud_args)
```

breakfast buffet	0.325662

It is interesting to see "bit worn" have a high positive PMI, since that usually is negative. But, this shows that customers are willing to accept things a bit worn and still give positive

reviews.

We see mention of "cornwal clean", which doesn't seem to have a direct meaning but is location specific.

We also see several phrases mention a jacuzzi, which shows the positive experience contributed by a good jacuzzi in the room.

```
pminegdf = pmiForAllCal(finaldfNP,topkNP,'negative').sort_values('pmi',ascending=0)
```

```
     100%|██████████| 1000/1000 [00:00<00:00, 1111.12it/s]
```

```
pminegdf[:50]
```

	pmi
pickup soft	1.280311
bathroom good	1.280311
updat u002fupgrad	1.280311
desk pictur u002fbathroom	1.280311
someth cold	1.280311
nmi biggest	1.280311
tub drain slow	1.280311
meet minimum	1.280311
updat real good	1.280311
smell bad	1.280311
wall ok	1.280311
good hot breakfast small	1.280311
wi fi work fine	1.280311

```
plotWordCloud(zip(pminegdf[:50].index, pminegdf[:50]['pmi'].values), **wordcloud_args)
```

'bathroom good' is unexpected to be part of a negative PMI, but it may show that customers aren't suitably pleased by just having a good bathroom.

'vegan cannot' shows up, it may show that not having vegan options is a dealbreaker for vegans that would result in negative reviews.

'meet minimum' may indicate that a hotel that meets the minimum requirements still may acheive a negative review.

```
top_df = df[df['hotelName']=='Auberge Chesley&#39;s Inn']
bot_df = df[df['hotelName']=='Howard Johnson by Wyndham Cornwall']
```

small lower	1.280311

```
topk1 = getTopKWords(top_df, 1000)
finaldf1 = dataFrameTransformation(df, topk1)[df['hotelName']=='Auberge Chesley&#39;s
topkNP1 = getTopKNP(top_df, 1000, grammar = grammar2)
finaldfNP1 = NPdataFrameTransformation(top_df, topkNP1, grammar2)
```

```
pmidf = pmiForAllCal(finaldf1,topk1,'positive').sort_values('pmi',ascending=0)
plotWordCloud(zip(pmidf[:50].index, pmidf[:50]['pmi'].values), **wordcloud_args)
```

```
100%|██████████| 1000/1000 [00:00<00:00, 2143.60it/s]
```

ny repli	1.280311

```
pmidf = pmiForAllCal(finaldf1,topk1,'negative').sort_values('pmi',ascending=0)
plotWordCloud(zip(pmidf[:50].index, pmidf[:50]['pmi'].values), **wordcloud_args)
```

```
100%|██████████| 1000/1000 [00:00<00:00, 2395.30it/s]
```


```
pmidf = pmiForAllCal(finaldfNP1,topkNP1,'positive').sort_values('pmi',ascending=0)
plotWordCloud(zip(pmidf[:50].index, pmidf[:50]['pmi'].values), **wordcloud_args)
```

```
100%|██████████| 260/260 [00:00<00:00, 2389.45it/s]
```


```
pmidf = pmiForAllCal(finaldfNP1,topkNP1,'negative').sort_values('pmi',ascending=0)
plotWordCloud(zip(pmidf[:50].index, pmidf[:50]['pmi'].values+min(pmidf[:50]['pmi'].val
```

```
100%|██████████| 260/260 [00:00<00:00, 1959.53it/s]
```


```
topk1 = getTopKWords(bot_df, 1000)
finaldf1 = dataFrameTransformation(df, topk1)[df['hotelName']=='Howard Johnson by Wynd
topkNP1 = getTopKNP(bot_df, 1000, grammar = grammar2)
finaldfNP1 = NPdataFrameTransformation(bot_df, topkNP1, grammar2)
```

```
pmidf = pmiForAllCal(finaldf1,topk1,'positive').sort_values('pmi',ascending=0)
plotWordCloud(zip(pmidf[:50].index, pmidf[:50]['pmi'].values), **wordcloud_args)
```

```
100%|██████████| 1000/1000 [00:00<00:00, 2240.25it/s]
```

```
pmidf = pmiForAllCal(finaldf1,topk1,'negative').sort_values('pmi',ascending=0)
plotWordCloud(zip(pmidf[:50].index, pmidf[:50]['pmi'].values), **wordcloud_args)
```

```
100%|██████████| 1000/1000 [00:00<00:00, 2078.41it/s]
```


```
pmidf = pmiForAllCal(finaldfNP1,topkNP1,'positive').sort_values('pmi',ascending=0)
plotWordCloud(zip(pmidf[:50].index, pmidf[:50]['pmi'].values+min(pmidf[:50]['pmi'].val
```

```
0%|          | 0/289 [00:00<?, ?it/s]100%|██████████| 289/289 [00:00<00:00, 2360
```


```
pmidf = pmiForAllCal(finaldfNP1,topkNP1,'negative').sort_values('pmi',ascending=0)
plotWordCloud(zip(pmidf[:50].index, pmidf[:50]['pmi'].values+min(pmidf[:50]['pmi'].val
```

```
100%|██████████| 289/289 [00:00<00:00, 2182.77it/s]
```

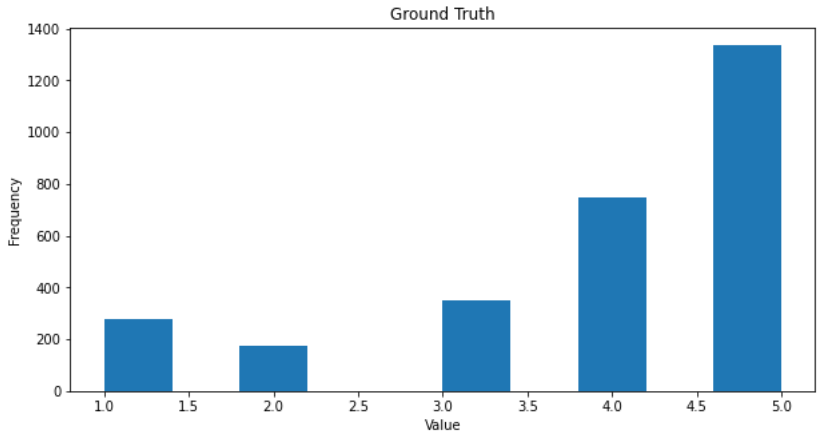
The best hotel seems to be modern, have nice rugs, and really amazing breakfast. The worst hotel seems to be dirty, have thin walls, mould and even blood stains. The grammar I selected does give good insight into the pros and cons of both hotels

Q5

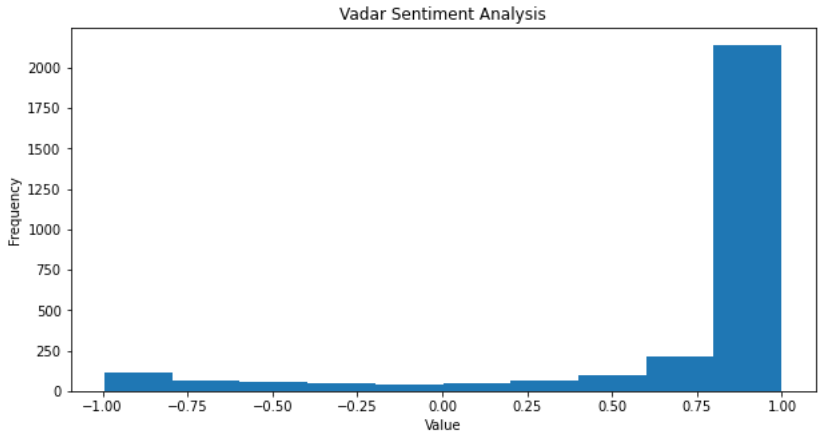
Note Remember to save a static image of the map in the notebook

```
def getHistogram(df, measure, title,figsize=(10,5)):
    fig = plt.figure(figsize=figsize)
    plt.title(title)

    if measure=='both':
        x = [df['ratingScore'].values/5]
        y = [df['vader'].values]
        bins = np.linspace(-1, 1, 100)
        plt.hist(x, bins, label='normalized Ground Truth')
        plt.hist(y, bins, label='vader')
        plt.legend(loc='upper right')
        plt.xlabel("Value")
        plt.ylabel("Frequency")
    else:
        plt.hist(df[measure].values)

    plt.xlabel("Value")
    plt.ylabel("Frequency")
```

```
getHistogram(df,'ratingScore', 'Ground Truth')
```

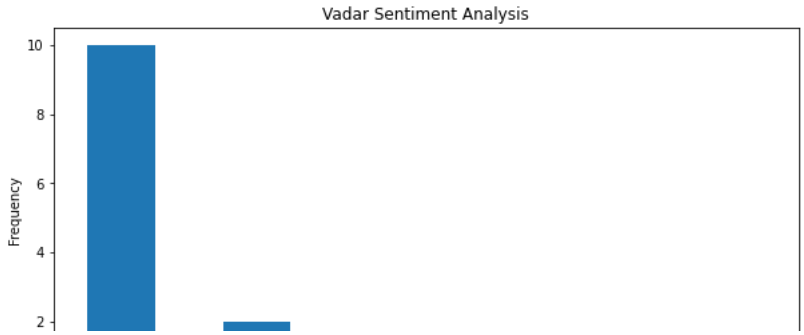



```
getHistogram(df, 'vader', 'Vadar Sentiment Analysis')
```



It is interesting that the Vader sentiment is so much more skewed to positive values. This is not surprising after thinking about it, since many reviews say good things about the hotel even if the overall experience was mediocre or negative; it's usually a few things that are very bad that lowers the rating. For example, saying something is "pretty good" might sound good but in terms of review score it can still be mediocre.

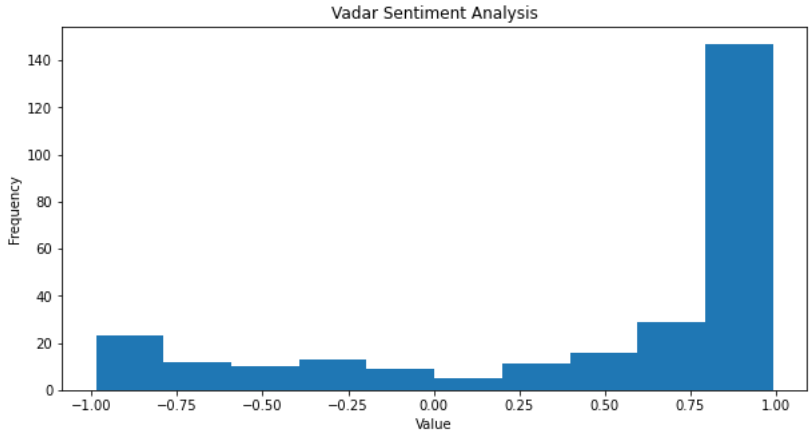
```
getHistogram(df.groupby('hotelName').count(), 'ratingScore', 'Vadar Sentiment Analysis
```



There appears to be a skew towards lower reviews, with only one place having over 1000 reviews.

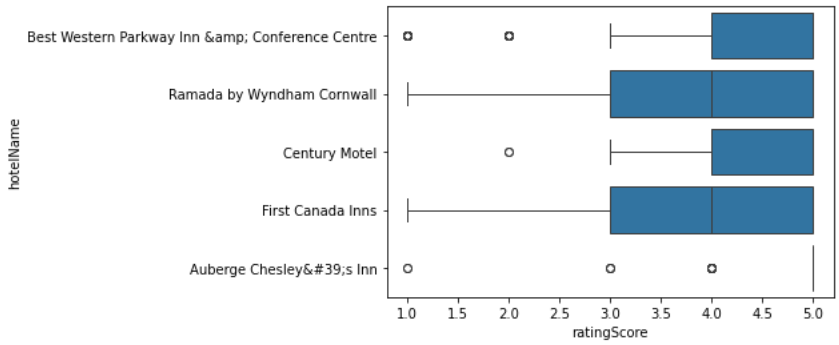
value

```
getHistogram(df[df['hotelName'] == 'Super 8 by Wyndham Cornwall ON'], 'vader', 'Vadar
```



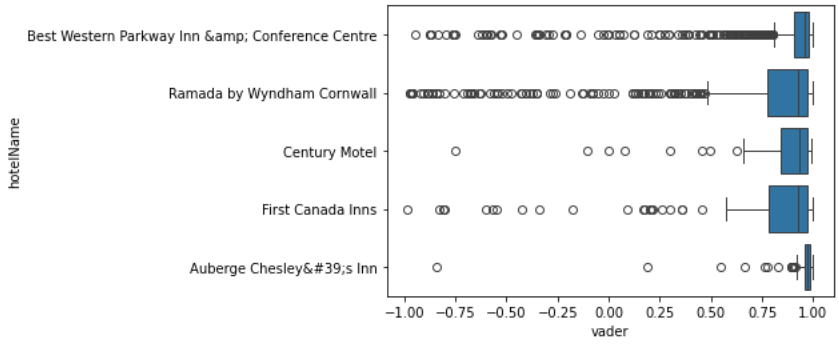

```
sns.boxplot(data=df[df['hotelName'].isin(true_sort[:5]['hotelName'])], y="hotelName",
```

```
<Axes: xlabel='ratingScore', ylabel='hotelName'>
```




```
sns.boxplot(data=df[df['hotelName'].isin(true_sort[:5]['hotelName'])], y="hotelName",
```

```
<Axes: xlabel='vader', ylabel='hotelName'>
```



```
df[df['hotelName'].isin(true_sort[:5]['hotelName'])]['ratingScore'].mean(), df[df['hot
```

```
(4.352542372881356, 0.924103271580607)
```

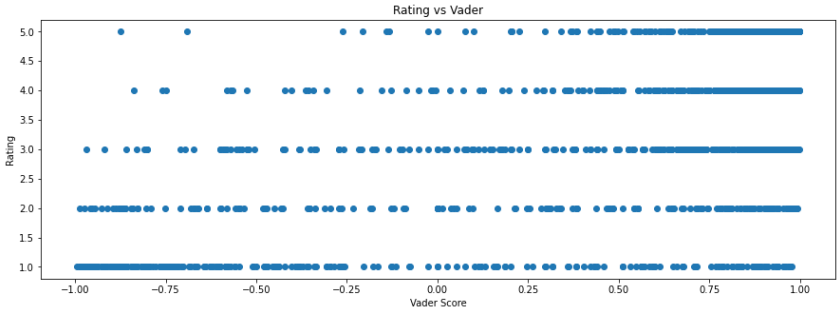
```
df[df['hotelName'].isin(true_sort[:5]['hotelName'])]['vader'].mean(), df[df['hotelName
```

```
(0.8262712832929782, 0.12420693025537756)
```

The boxplots are much more informative, since they show the distribution of the data better. The data is skewed towards the higher scores for both metrics, and thus the mean is much lower than the median. the variance is also harder to interpret for this non-normal data distribution.

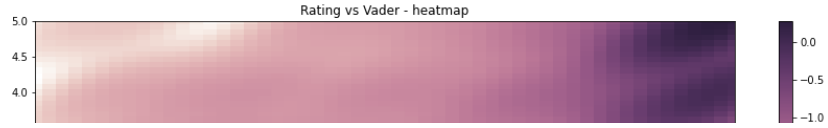
```
fig, ax = plt.subplots(1,1,figsize=(15,5),sharex=False,sharey=False)
rating_scores = df['ratingScore'].values
vader_scores = df['vader'].values
ax.plot(vader_scores, rating_scores,"o")

ax.set_title('Rating vs Vader')
ax.set_ylabel('Rating')
ax.set_xlabel('Vader Score')
```

```
Text(0.5, 0, 'Vader Score')
```




```
k = gaussian_kde(np.vstack([vader_scores, rating_scores]))
xi, yi = np.mgrid[vader_scores.min():vader_scores.max():vader_scores.size**0.5*1j,rati
zi = k(np.vstack([xi.flatten(), yi.flatten()]))
cmap = sns.cubehelix_palette(light=1, as_cmap=True)
fig, ax1 = plt.subplots(1,1, figsize=(15,5))


b1 = ax1.pcolormesh(xi, yi, np.log10(zi.reshape(xi.shape)), cmap=cmap)

ax1.set_xlim(vader_scores.min(), vader_scores.max())
ax1.set_ylim(rating_scores.min(), rating_scores.max())

ax1.set_xlabel('Vader Score')
ax1.set_ylabel('Rating')

fig.colorbar(b1, ax=ax1)

ax1.set_title('Rating vs Vader - heatmap')
```

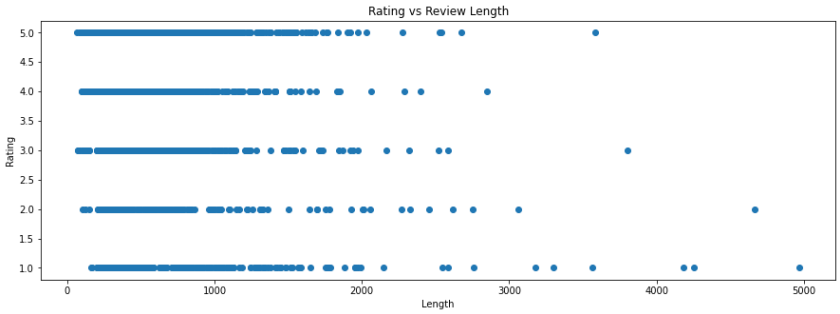
```
Text(0.5, 1.0, 'Rating vs Vader - heatmap')
```



I can vaguely see that the lower Vader scores are correlated with lower Ratings, since there is a high density of data at -1, 1 and another high density at 1, 5. This shows that Vader score does have some correlation to true ratings, but it's far from perfect

```
fig, ax = plt.subplots(1,1,figsize=(15,5),sharex=False,sharey=False)
#rating_scores = df['ratingScore'].values
#vader_scores = df['vader'].values
review_len = np.array([len(i) for i in df['reviewColumn'].values])

ax.plot(review_len, rating_scores,"o")

ax.set_title('Rating vs Review Length')
ax.set_ylabel('Rating')
ax.set_xlabel('Length')
```

```
Text(0.5, 0, 'Length')
```

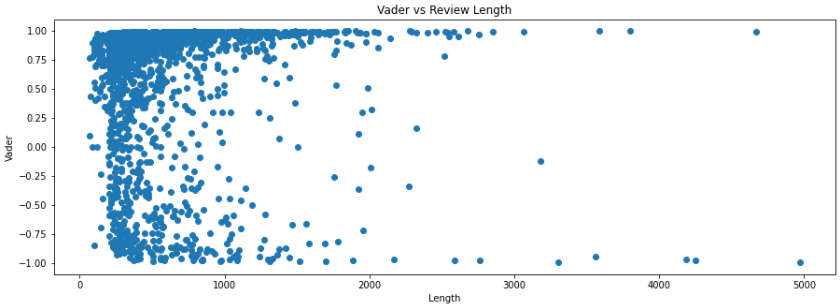



```
k = gaussian_kde(np.vstack([review_len, rating_scores]))
xi, yi = np.mgrid[review_len.min():review_len.max():review_len.size**0.5*1j,rating_sco
zi = k(np.vstack([xi.flatten(), yi.flatten()]))
cmap = sns.cubehelix_palette(light=1, as_cmap=True)
fig, ax1 = plt.subplots(1,1, figsize=(15,5))


b1 = ax1.pcolormesh(xi, yi, np.log10(zi.reshape(xi.shape)), cmap=cmap)

ax1.set_xlim(review_len.min(), review_len.max())
ax1.set_ylim(rating_scores.min(), rating_scores.max())

ax1.set_xlabel('Review Length')
ax1.set_ylabel('Rating')

fig.colorbar(b1, ax=ax1)

ax1.set_title('Rating vs Review Length - heatmap')
```

```
Text(0.5, 1.0, 'Rating vs Review Length - heatmap')
```

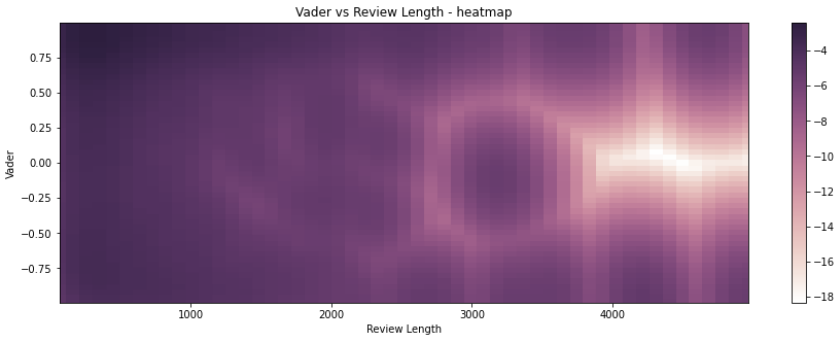

```
fig, ax = plt.subplots(1,1,figsize=(15,5),sharex=False,sharey=False)
review_len = np.array([len(i) for i in df['reviewColumn'].values])

ax.plot(review_len, vader_scores,"o")

ax.set_title('Vader vs Review Length')
ax.set_ylabel('Vader')
ax.set_xlabel('Length')
```

```
Text(0.5, 0, 'Length')
```




```
k = gaussian_kde(np.vstack([review_len, vader_scores]))
xi, yi = np.mgrid[review_len.min():review_len.max():review_len.size**0.5*1j,vader_scor
zi = k(np.vstack([xi.flatten(), yi.flatten()]))
cmap = sns.cubehelix_palette(light=1, as_cmap=True)
fig, ax1 = plt.subplots(1,1, figsize=(15,5))


b1 = ax1.pcolormesh(xi, yi, np.log10(zi.reshape(xi.shape)), cmap=cmap)

ax1.set_xlim(review_len.min(), review_len.max())
ax1.set_ylim(vader_scores.min(), vader_scores.max())

ax1.set_xlabel('Review Length')
ax1.set_ylabel('Vader')

fig.colorbar(b1, ax=ax1)

ax1.set_title('Vader vs Review Length - heatmap')
```

```
Text(0.5, 1.0, 'Vader vs Review Length - heatmap')
```

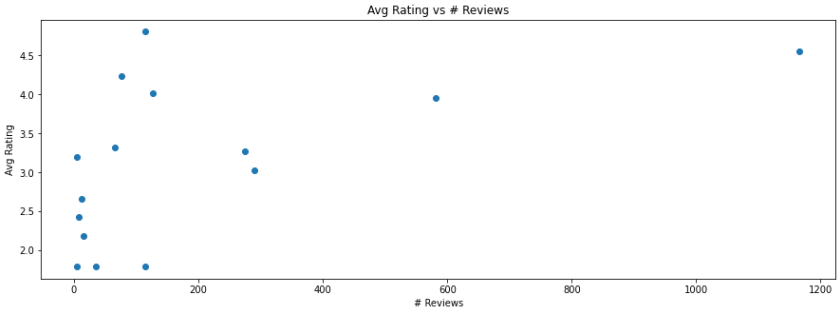


There is a slight U-shape to the data: the longer reviews tend to be more extreme towards either positive or negative scores.

This trend appears much more visibly in the Vader scores than the ratings.

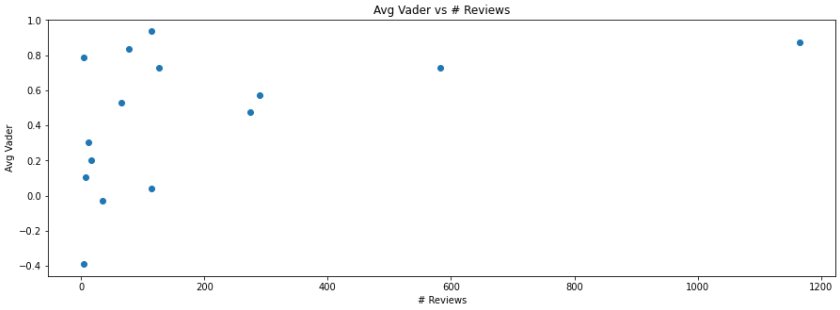
```
fig, ax = plt.subplots(1,1,figsize=(15,5),sharex=False,sharey=False)
review_len = np.array([len(i) for i in df['reviewColumn'].values])

ax.plot(df.groupby('hotelName').count()['ratingScore'].values, df.groupby('hotelName')

ax.set_title('Avg Rating vs # Reviews')
ax.set_ylabel('Avg Rating')
ax.set_xlabel('# Reviews')
```

```
C:\Users\kirby\AppData\Local\Temp/ipykernel_49616/4129118619.py:4: FutureWarning:
  ax.plot(df.groupby('hotelName').count()['ratingScore'].values, df.groupby('hotel
Text(0.5, 0, '# Reviews')
```




```
fig, ax = plt.subplots(1,1,figsize=(15,5),sharex=False,sharey=False)
review_len = np.array([len(i) for i in df['reviewColumn'].values])

ax.plot(df.groupby('hotelName').count()['ratingScore'].values, df.groupby('hotelName')

ax.set_title('Avg Vader vs # Reviews')
ax.set_ylabel('Avg Vader')
ax.set_xlabel('# Reviews')
```

```
C:\Users\kirby\AppData\Local\Temp/ipykernel_49616/2829231197.py:4: FutureWarning:
  ax.plot(df.groupby('hotelName').count()['ratingScore'].values, df.groupby('hotel
Text(0.5, 0, '# Reviews')
```



More reviews typically have high scores for both rating and vader.

However, it is possible to have high scores with low # reviews, but a low # of ratings only has low scoring hotels.

```
def make_address(row):
      # individual columns need to be combined
    print(row['streetAddress'], row['City'], row['Province'])
    return row['streetAddress']+", "+row['City']+", "+row['Province']
```

```
hotel_scores.drop(columns=['hotelName'])
```

	mean ground truth	mean vader compound	median vader compound
0	4.807018	0.935478	0.97300
1	4.548885	0.874945	0.95730
2	4.233766	0.834825	0.93160
3	3.031034	0.574015	0.89950
4	3.323077	0.529631	0.83600
5	4.015873	0.729245	0.92585
6	1.789474	0.041847	0.03870
7	2.187500	0.201037	0.45765
8	2.666667	0.303600	0.37765
9	1.800000	-0.029906	-0.36370
10	2.428571	0.105414	0.79600
11	1.800000	-0.390300	-0.53460
12	3.958763	0.727240	0.92645
13	3.200000	0.786460	0.97800
14	3.276364	0.476064	0.83920

```
# Need the location for each hotel in a format the tool can handle

geo_rating_df = hotel_scores.drop(columns=['hotelName']).join(df.groupby('hotelName', 
geo_rating_df['formed_address'] = geo_rating_df.apply(make_address , axis=1)
geo_rating_df = geo_rating_df.replace(["6032 Lundy's Lane, Niagara Falls, Ontario"],'7
geo_rating_df = geo_rating_df.head()
```

```
40 First St W ('Cornwall',) Ontario
1515 Vincent Massey Dr ('Cornwall',) Ontario
1209 Brookdale Ave ('Cornwall',) Ontario
1625 Vincent Massey Dr Rr 2 ('Cornwall',) Ontario
1123 Brookdale Ave ('Cornwall',) Ontario
1618 Vincent Massey Dr ('Cornwall',) Ontario
1142 Brookdale Ave ('Cornwall',) Ontario
1750 Vincent Massey Dr ('Cornwall',) Ontario
2200 Vincent Massey Dr ('Cornwall',) Ontario
1700 Montreal Rd ('Cornwall',) Ontario
2120 Vincent Massey Dr ('Cornwall',) Ontario
1618 Vincent Massey Dr ('Cornwall',) Ontario
805 Brookdale Ave ('Cornwall',) Ontario
1620 Vincent Massey Dr ('Cornwall',) Ontario
2694 Brookdale Ave ('Cornwall',) Ontario
```

```
# 0 - need to give the tool a generic name.
locator = Photon(user_agent='myGeocoder')
# 1 - conveneint function to delay between geocoding calls
geocode = RateLimiter(locator.geocode, min_delay_seconds=1)
# 2 - form the location string
geo_rating_df['location'] = geo_rating_df['formed_address'].apply(geocode)
# 3 - create longitude, laatitude and altitude from location column (returns tuple)
geo_rating_df['point'] = geo_rating_df['location'].apply(lambda loc: tuple(loc.point)
# 4 - split point column into latitude, longitude and altitude columns
geo_rating_df[['latitude', 'longitude', 'altitude']] = pd.DataFrame(geo_rating_df['poi

geo_rating_df.head()
```

	mean ground truth	mean vader compound	median vader compound	hotelName	streetAddress	City	Province