UNIVERSITI MALAYA

WIA2005
ALGORITHM DESIGN AND ANALYSIS

Semester 2 Session 2021/2022

Project Report

Lecturer: DR. ADELEH ASEMI ZAVAREH

Tutorial 2 Group 5

Name	Matric Number
Lawrence Leroy Chieng Tze Yao	S2018935
Kang Zhi Hui	U2005282
Hong Zhao Cheng	U2005280
Lim Jing Ying	U2005397
Chai Yu Xuan	U2005334
Yau De Min	U2005347

Table of Contents

Introduction **3**

Description **4**

Problem 1 4

Problem Description 4

Tools and Algorithms 4

Library 4

Algorithm Description 5

Time Complexity 6

Algorithmic Conclusion 6

Source Code 8

Snapshots of Input/Output 12

Problem 2 23

Problem Description 23

Tools and Algorithms 23

API (Bing Map Distance Matrix API) 23

Library 24

Algorithm Description 25

Time Complexity 26

Source Code 27

Snapshots of Input/Output 32

Problem 3 37

Problem Description 37

Tools and Algorithms 37

Library 37

Time Complexity 38

Source Code 38

SnapShot Input/Output 40

Conclusion **41**

References **42**

Appendices **47**

Introduction

Recently, many companies are getting interested in expansion of business as it is actually a beneficial way to reach out to more types of customers. In the project brief, we know that Moonbucks is a coffee chain that has stores located all over the world. The company is constantly looking at running better logistics as well as expansion to open more stores at strategic locations. Our group decided to tackle this question by analyzing and providing insights to the management for making business decisions. We have been given a sample dataset that contains the location of Moonbucks stores all over the world to **determine the suitable country** and **location of stores** to expand its business. The analysation will be further explained in this report.

Description

Problem 1

Problem Description

Moonbucks intends to develop their business by opening more locations around the world. We evaluate the local economic and social factors in the chosen countries to assure maximum earnings. These countries are United Arab Emirates (AE), Malaysia (MY), Singapore (SG), United Kingdom (UK), and United States (US). We search for five articles on internet news sources that contain content related to local economy and social situation in each country. Before we analyze articles for each country, we convert these online articles to text through textise website. Then, we implement Brute Force Algorithms to calculate words count for positive, negative and neutral words in articles while filtering out stop words in order to ensure the accuracy of results. To gain insight into the local economic and social condition, we do a sentiment analysis of the article's positive, negative, and neutral words. Logically, we assume the more positive words extracted, the better the country's financial situation will be.

Tools and Algorithms

Library

Pandas is a software library written for the Python programming language that our team used to design the solution. We imported pandas as pd in Python code and utilized the DataFrame structure that consists of three components: rows, columns, and data. The Pandas DataFrame is a tabular data structure with rows and columns that is two-dimensional, size-mutable, and possibly heterogeneous. The basic operations which can be performed on Pandas DataFrame are creating a DataFrame, dealing with rows and columns, indexing and selecting data, working with missing data and iterating over rows and columns. We stored sorted data in Pandas DataFrame then these data are exported to CSV files using Pandas .to_csv for better visualization.

Plotly is a graphing library that makes interactive and publication-quality graphs. We use Plotly to customize bar charts for displaying word frequency and word count of positive and negative words for each country. For grouped bar charts, we apply subplots to make it convenient to annotate bars with labels. For bar charts showing word count of positive and negative words, we also use the bar function to result in one rectangle drawn per row of input and control the text labels with uniformtext layout parameters.

Algorithm Description

Our team implemented Brute Force Algorithms to compute word count for positive, negative, and neutral words in all articles. The Brute Force Algorithm is an intuitive and simple problem-solving technique that iterates all possible solutions rather than eliminating some, like other smart algorithms do. It is not restricted to any particular problem domain and is best suited to handling minor and simple problems. In contrast, the Brute Force approach is slow and inefficient.

After we processed all online articles into text files, we read these files from our local computer. All sentences are then lowercase and splited into a list of words. We divided the words into three categories: good words, bad words, and neutral words in arrays after filtering all stop words from articles. The word frequency for each type of word array was then calculated. We chose only the good and bad word arrays for sentiment analysis, then put the data in Pandas DataFrame with two columns entitled "Word" and "Frequency." After that, we wrote the tabular data into CSV files and printed the total positive, negative, and neutral words.

Time Complexity

Filter stopwords: **O(mn)**,

where m = len(stopwords),

and n = len(article).

Sort article into good, bad and neutral array: **O(mn)**,

where m = [len(goodwords) + len(badwords)] / 2 in average case,

and n = len(article_without_stopwords)

Creating word-frequency dictionary: **O(n)**,

where n = len(word_array)

Hence, the total average time complexity for **one** article is O[(2m+1)n] **= O(mn)**

Algorithmic Conclusion

A sentiment analysis helps Moonbucks to figure out the positivity and negativity of each potential country for expanding their business by adding the number of stores around the world. To analyze the local economic and social situation of the five countries which are United Arab Emirates (AE), Malaysia (MY), Singapore (SG), United Kingdom (UK), and United States (US), first we searched five articles related to the country's local economic and social situation for all the five countries, then we extract usable words which are positive words, negative words and neutral words from articles for sentiment analysis. In the meantime, we filter out stop words from articles we found to obtain a more accurate result.

The string-matching algorithm approach employed by our team is Brute Force Algorithms to analyze the positive words, negative words, and neutral words in all articles. Brute Force Algorithms try every possibility rather than eliminating many possible solutions which are inefficient in comparison to other advanced algorithms. On the other hand, Brute Force Algorithm is a straightforward problem-solving strategy that always guarantees the correct solution.

Country	Positive		Negative		Total
	Word	Percentage (%)	Word	Percentage (%)	
AE	446	67.68	213	32.32	659
MY	182	72.80	68	27.2	250
SG	95	64.19	53	35.81	148

UK	**103**	**50.99**	**99**	**49.01**	**202**
US	**160**	**64.78**	**87**	**35.22**	**247**
Total	**986**		**520**		**1506**

Table 1: Positive and negative words count for 5 countries using Brute Force Algorithms

As shown in Table 1, we can conclude that in general all five countries which are AE, MY, SG, UK, and US have more positive words in articles thus they are giving positive sentiment. Among all the countries, Malaysia shows the highest percentage (72.80%) of positive words while the United Kingdom shows the lowest percentage (50.99%) of positive words. The United Arab Emirates ranks second by showing 67.68% of positive words. On the other hand, the United States and Singapore show a similar percentage of positive words which are 64.78% and 64.19%.

Based on the sentiment analysis of Malaysia's local economic and social situation articles, positive words like growth, strength, sustained, improvement, astonishing, confidence, leading, achieve and just to name a few were being extracted. As Malaysia has the highest rate of having positive words, we conclude Malaysia is the first choice for Moonbucks to expand its branch, followed by AE, US, SG and UK.

Source Code

```
# Problem 1
def filter_stopwords(text=[]):
  return list(filter(lambda t: t not in stopwords, text))


def word_freq(text=[]):
  return sorted(list(set(zip(text, list(map(lambda t: text.count(t), text))))), key=lambda x: x[1],
reverse=True)


def sort_words(text=[]):
  good = []
  bad = []
  neutral = []

  for t in text:
    if t in goodwords:
      good.append(t)
    elif t in badwords:
      bad.append(t)
    else:
      neutral.append(t)

  return good, bad, neutral


def count_words(text=[]):
  return sum(list(map(lambda t: t[1], text)))


# Plot Graph
def plotChart(country):
  df_for_plotting = pd.read_csv(f'csv/{country}.csv')

  plotGoodWords(country)
  plotBadWords(country)
  plotWordCountPerCountryArticle(df_for_plotting.head(5))


def plotBadWords(state):
  filename = 'csv/bad-' + state + '.csv'
 df = pd.read_csv(filename)

 # each state
 countries = {
   "AE": "United Arab Emirates",
   "MY": "Malaysia",
   "SG": "Singapore",
   "US": "United States",
   "UK": "United Kingdom"
 }
 title = "Frequency of Bad Words for " + countries[state]
```

```
    fig = px.bar(df, y='Frequency', x='Word', text_auto='.2d', title=title)
    fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
    fig.show()


def plotGoodWords(state):
  filename = 'csv/good-' + state + '.csv'
  df = pd.read_csv(filename)

# each state
  countries = {
   "AE": "United Arab Emirates",
   "MY": "Malaysia",
   "SG": "Singapore",
   "US": "United States",
   "UK": "United Kingdom"
 }
  title = "Frequency of Bad Words for " + countries[state]


    fig = px.bar(df, y='Frequency', x='Word', text_auto='.2d', title=title)
    fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
    fig.show()


def plotWordCountPerCountryArticle(df):
    x = np.arange(5)
    width = 0.35

    fig, ax = plt.subplots()
    rects1 = ax.bar(x - width / 2, df['positive_word'].tolist(), width, label='Positive Words')
    rects2 = ax.bar(x + width / 2, df['negative_word'].tolist(), width, label='Negative Words')

    ax.set_ylabel('Words Count')
    ax.set_title('Number of Word Count for 5 articles per country')
    ax.set_xticks(x, df['Article'].tolist())
    ax.legend()

    ax.bar_label(rects1, padding=3)
    ax.bar_label(rects2, padding=3)

    fig.tight_layout()

    plt.show()


def plotOverview():
    Articles = ["AE", "MY", "SG", "UK", "US"]
    positive = []
    negative = []
    for i in Articles:
        df = pd.read_csv(f'csv/{i}.csv')
        positive.append(df['positive_word'].iloc[5])
        negative.append(df['negative_word'].iloc[5])
```

```
    x = np.arange(len(Articles))
    width = 0.35

    fig, ax = plt.subplots()
    rects1 = ax.bar(x - width / 2, positive, width, label='Positive Words')
    rects2 = ax.bar(x + width / 2, negative, width, label='Negative Words')

    ax.set_ylabel('Words Count')
    ax.set_title('Number of Word Count for 5 countries')
    ax.set_xticks(x, Articles)
    ax.legend()

    ax.bar_label(rects1, padding=3)
    ax.bar_label(rects2, padding=3)

    fig.tight_layout()

    plt.show()


while True:
    filename = input("\nEnter filename (or type '...' to exit): ").upper()
    if filename == "...":
        break
    file = "Text/" + filename + ".txt"

    try:
        with open(file, "r", encoding="utf8") as f:
            content = f.read().lower().split()
            content = filter_stopwords(content)
            good_arr, bad_arr, neutral_arr = sort_words(content)

            good_arr = word_freq(good_arr)
            bad_arr = word_freq(bad_arr)
            neutral_arr = word_freq(neutral_arr)

            good_df = pd.DataFrame(good_arr, columns=['Word', 'Frequency'])
            bad_df = pd.DataFrame(bad_arr, columns=['Word', 'Frequency'])

            good_df.to_csv("csv/good-" + filename + ".csv", index=False)
            bad_df.to_csv("csv/bad-" + filename + ".csv", index=False)

            tot_good = count_words(good_arr)
            tot_bad = count_words(bad_arr)
            tot_neut = count_words(neutral_arr)

            csv_file = "csv/" + filename[:2] + ".csv"
            overview_file = "csv/Overview.csv"
            data = {
                'article': [filename],
                'positive_words': [tot_good],
                'negative_words': [tot_bad]
            }
```

```
            df = pd.DataFrame(data)
            df.to_csv(csv_file, mode='a', index=False, header=False)
            df.to_csv(overview_file, mode='a', index=False, header=False)

            print("File: %s\nTotal positive words: %d\nTotal negative words: %d\nTotal neutral words: %d\n"
                  % (filename, tot_good, tot_bad, tot_neut))

    except IOError as e:
        print("Error{0}: {1}".format(e.errno, e.strerror))

YesOrNo = int(input("Enter your 1 for plotting or 0 to continue the program: "))
if YesOrNo == 1:
    countries = ["AE", "MY", "SG", "UK", "US"]
    for i in countries:
        plotChart(i)

    # Plot overview
    plotOverview()
```

Snapshots of Input/Output

```
Enter filename (or type '...' to exit): AE
File: AE
Total positive words: 446
Total negative words: 213
Total neutral words: 7009


Enter filename (or type '...' to exit): MY
File: MY
Total positive words: 182
Total negative words: 68
Total neutral words: 3080


Enter filename (or type '...' to exit): SG
File: SG
Total positive words: 95
Total negative words: 52
Total neutral words: 1516
```

```
Enter filename (or type '...' to exit): UK
File: UK
Total positive words: 103
Total negative words: 99
Total neutral words: 2255


Enter filename (or type '...' to exit): US
File: US
Total positive words: 160
Total negative words: 87
Total neutral words: 3308
```

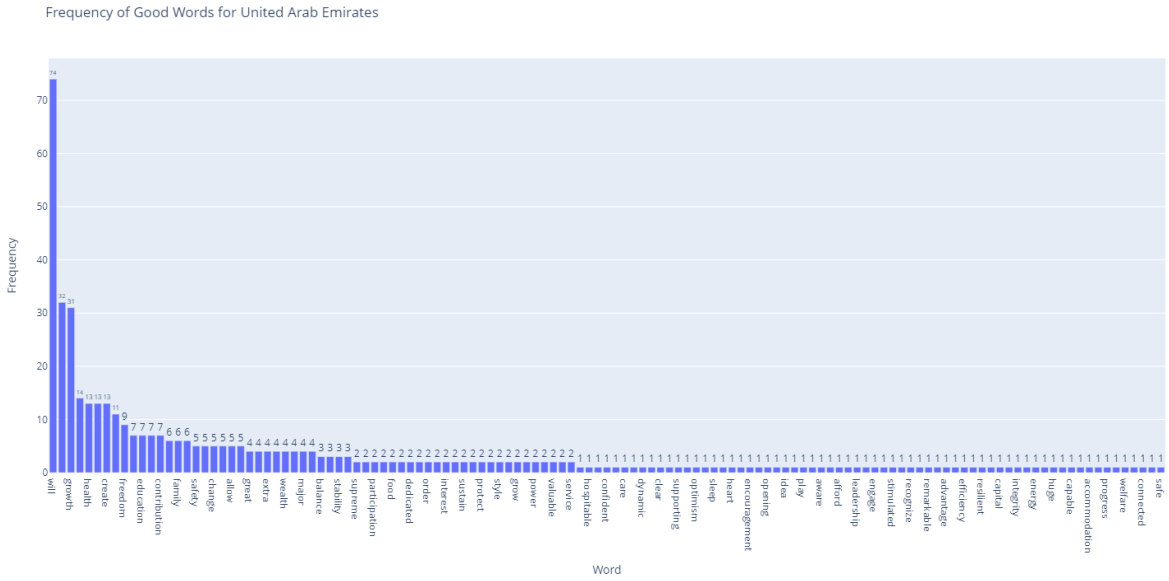


Figure 1.1: Total positive word count for 5 articles of United Arab Emirates

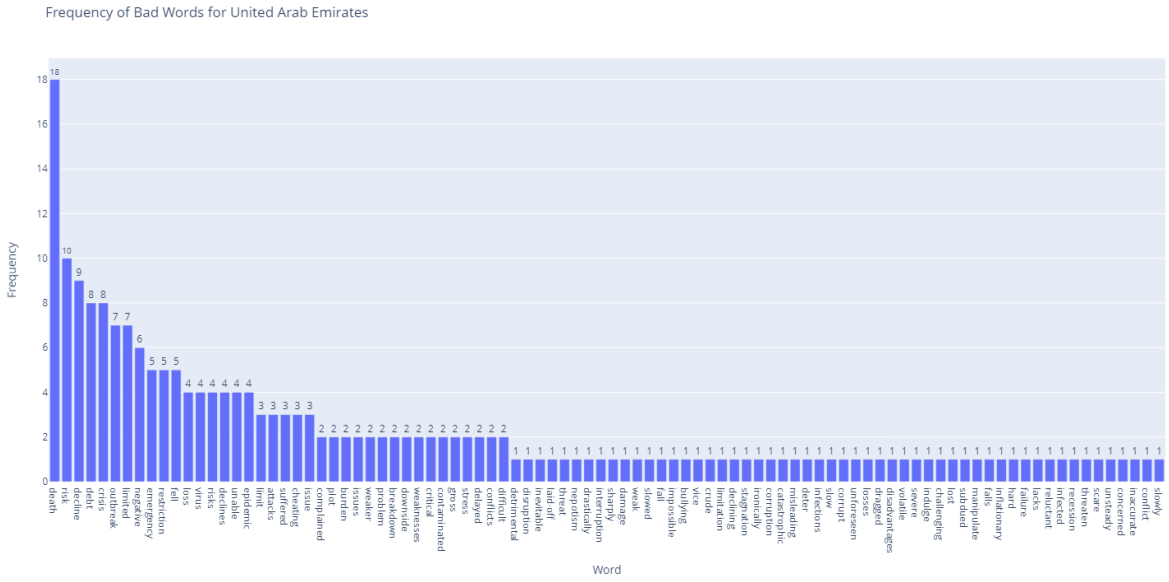


Figure 1.2: Total negative word count for 5 articles of United Arab Emirates

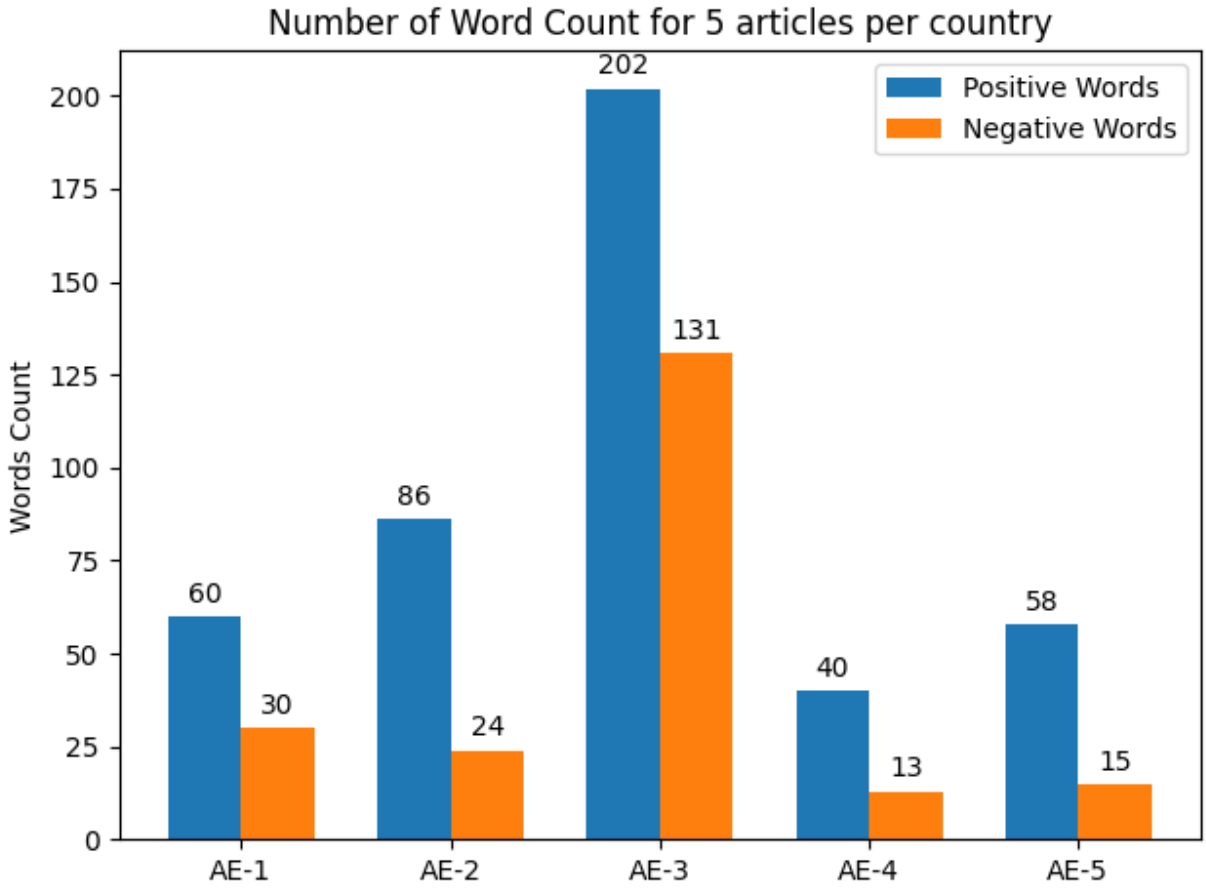


Figure 1.3: Total positive and negative word count for 5 articles of United Arab Emirates

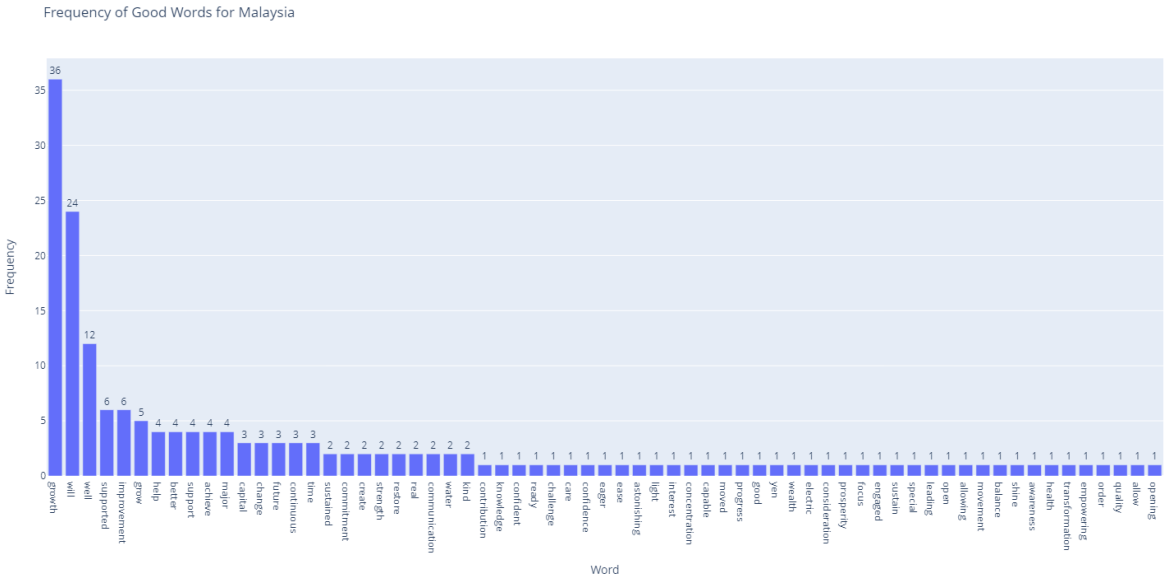


Figure 1.4: Total positive word count for 5 articles of Malaysia

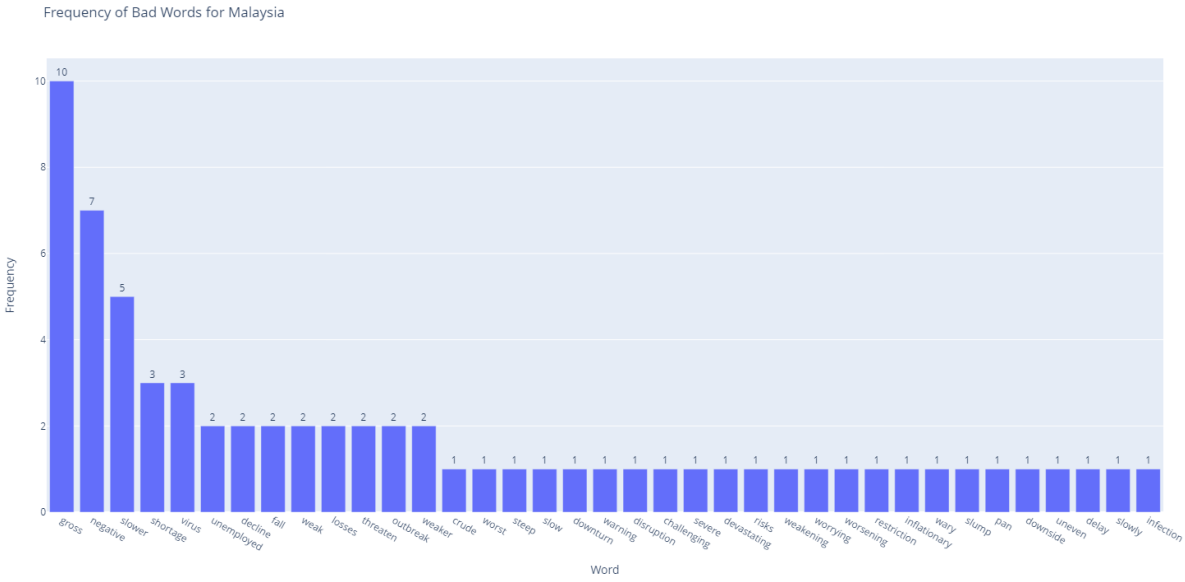


Figure 1.5: Total negative word count for 5 articles of Malaysia

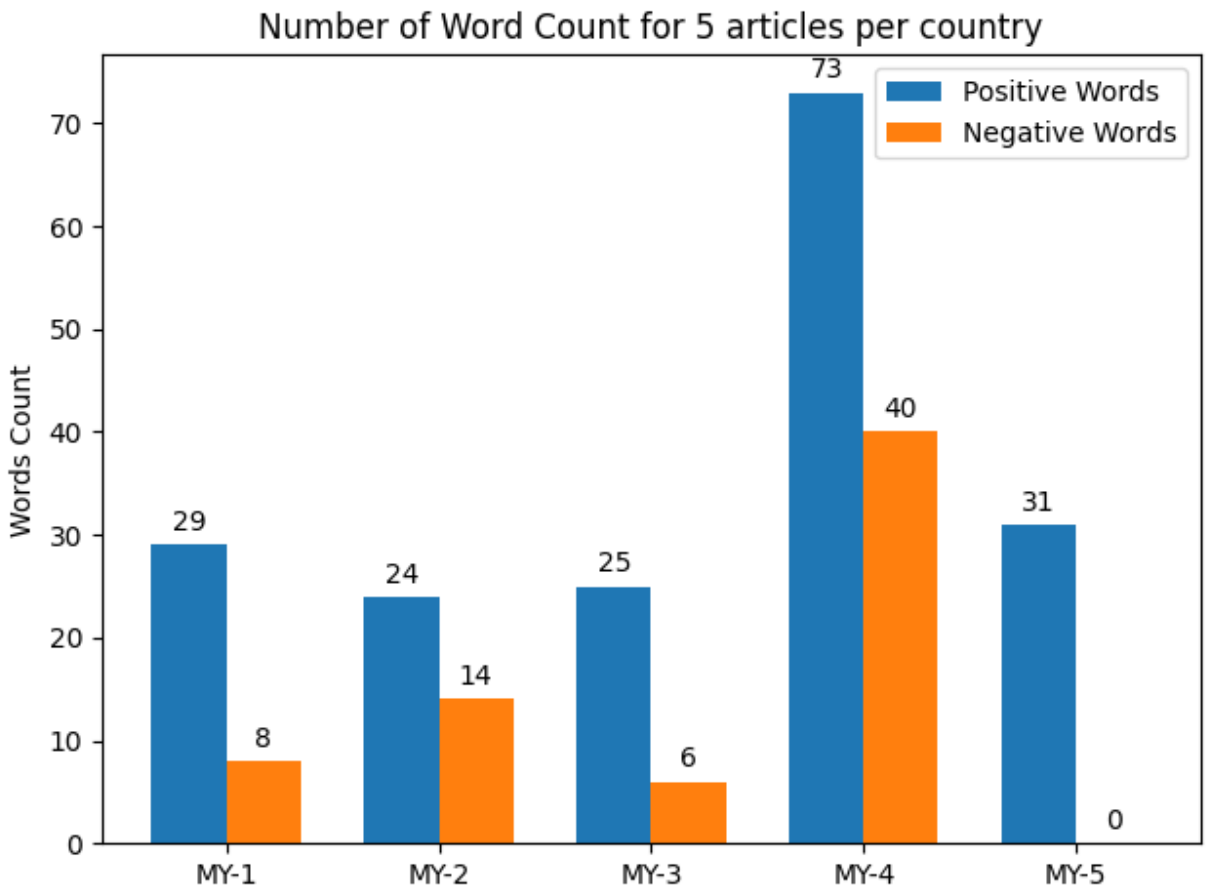


Figure 1.6: Total positive and negative word count for 5 articles of Malaysia

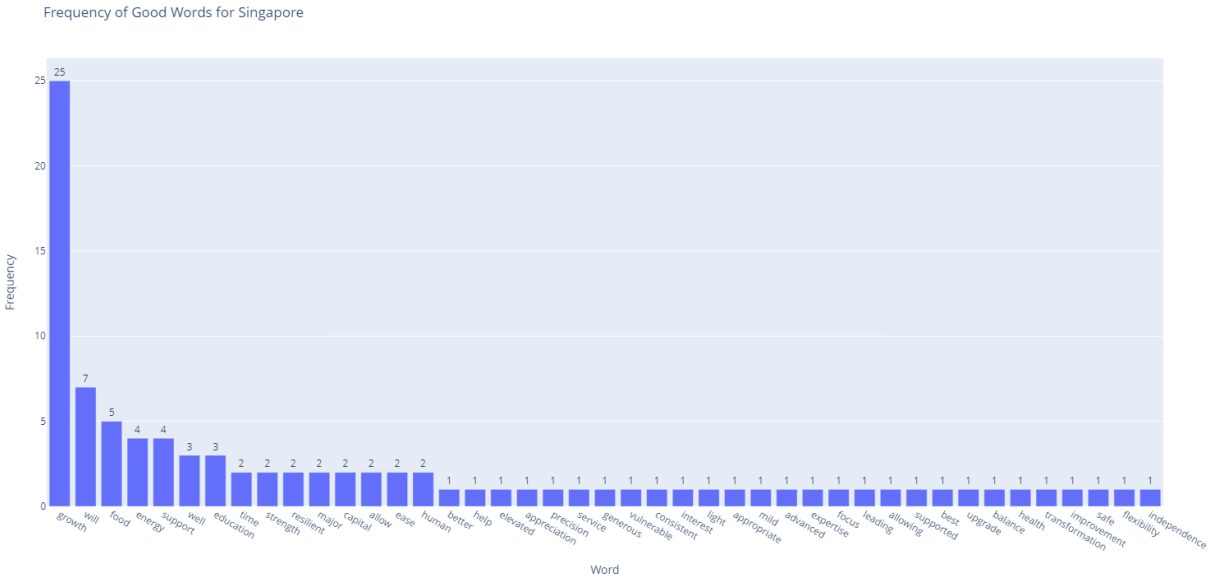


Figure 1.7: Total positive word count for 5 articles of Singapore

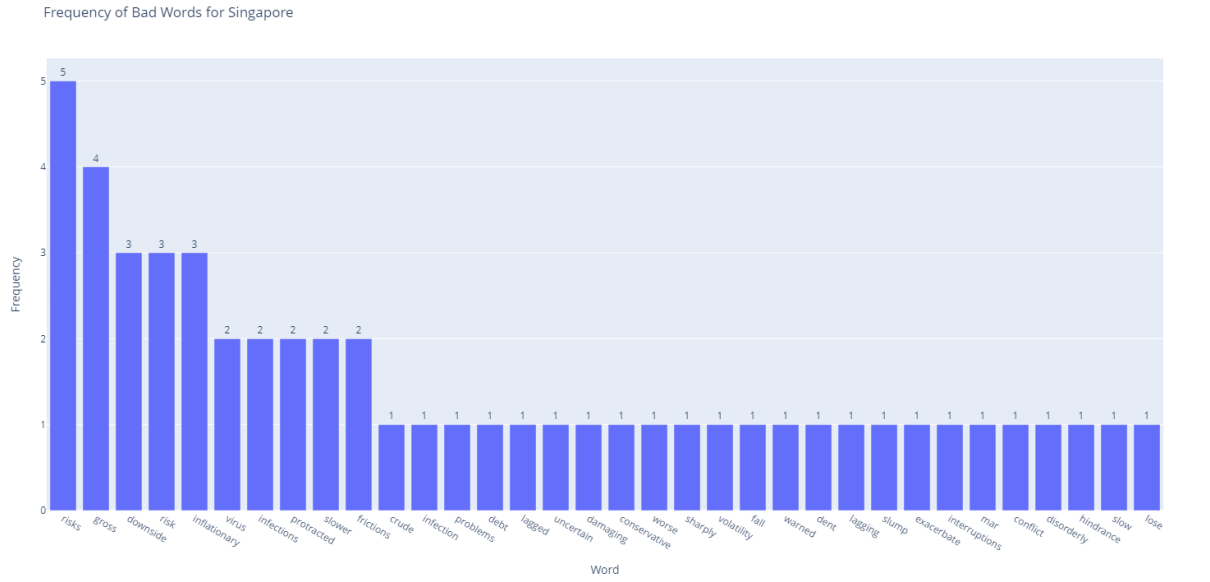


Figure 1.8: Total negative word count for 5 articles of Singapore

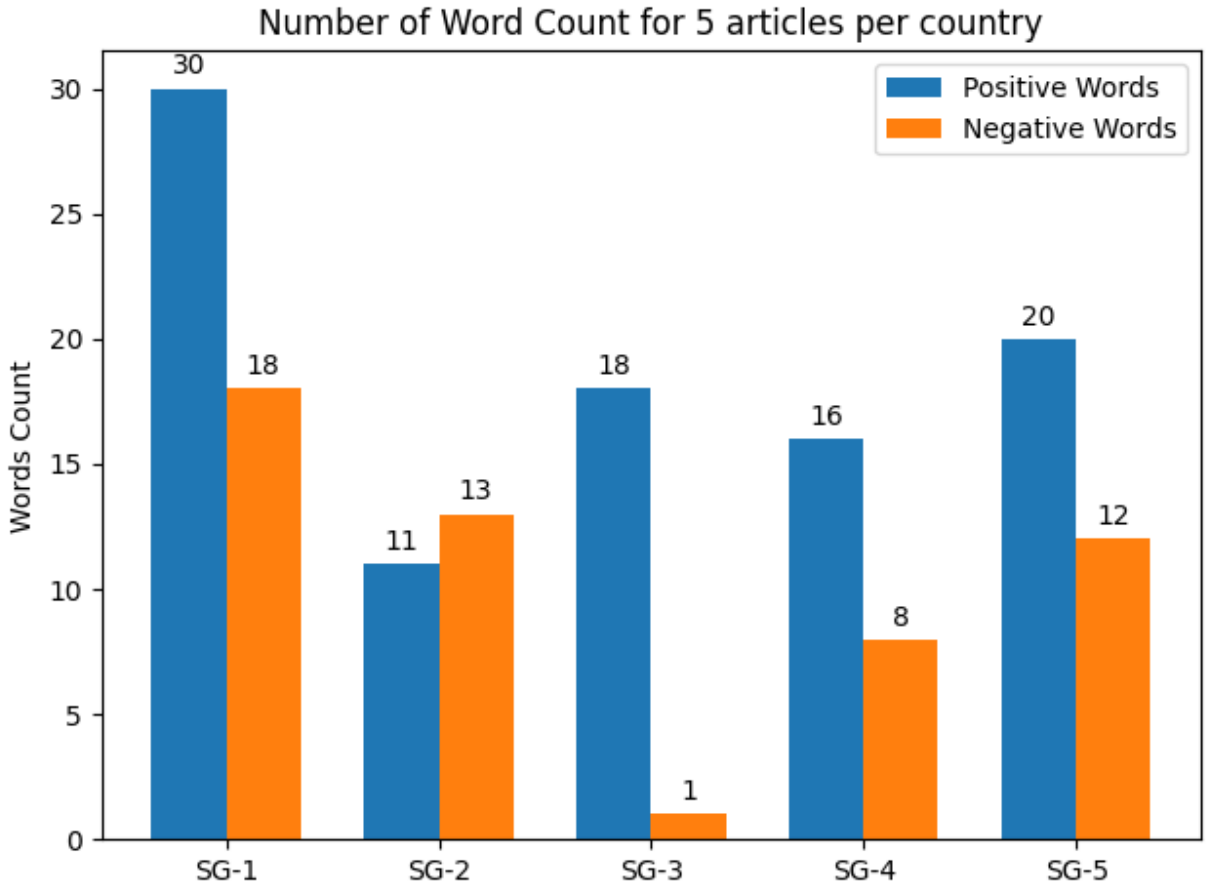


Figure 1.9: Total positive and negative word count for 5 articles of Singapore

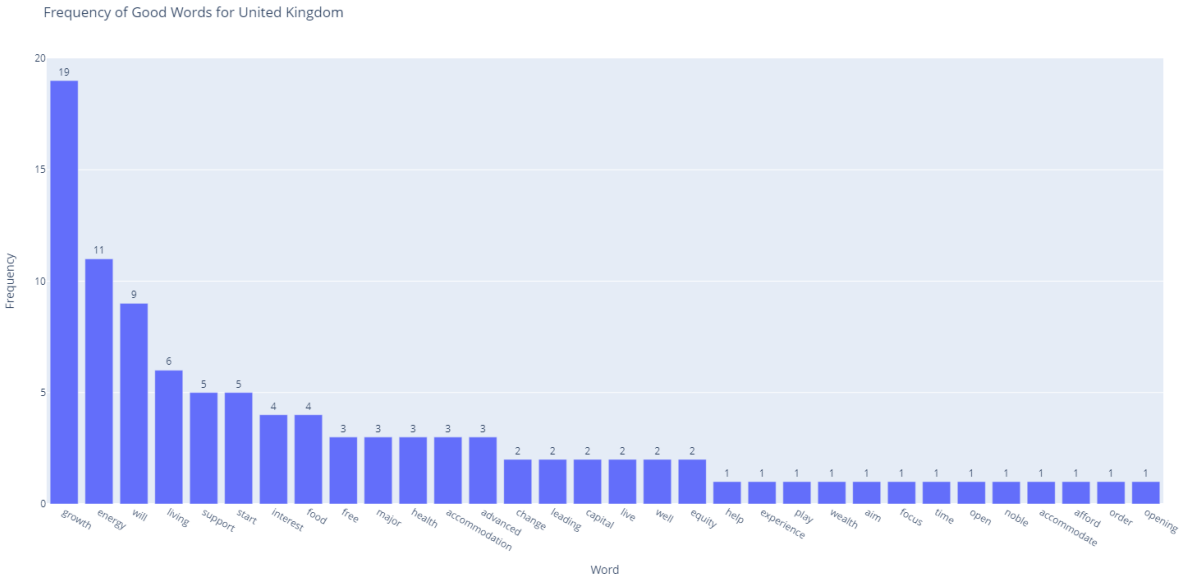


Figure 1.10: Total positive word count for 5 articles of United Kingdom

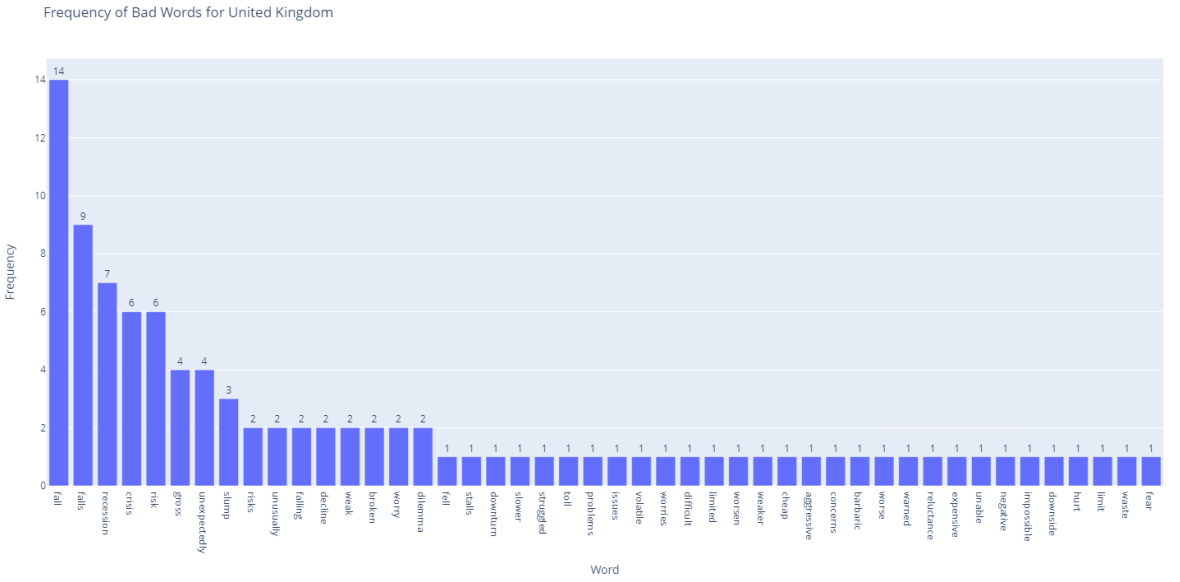


Figure 1.11: Total negative word count for 5 articles of United Kingdom

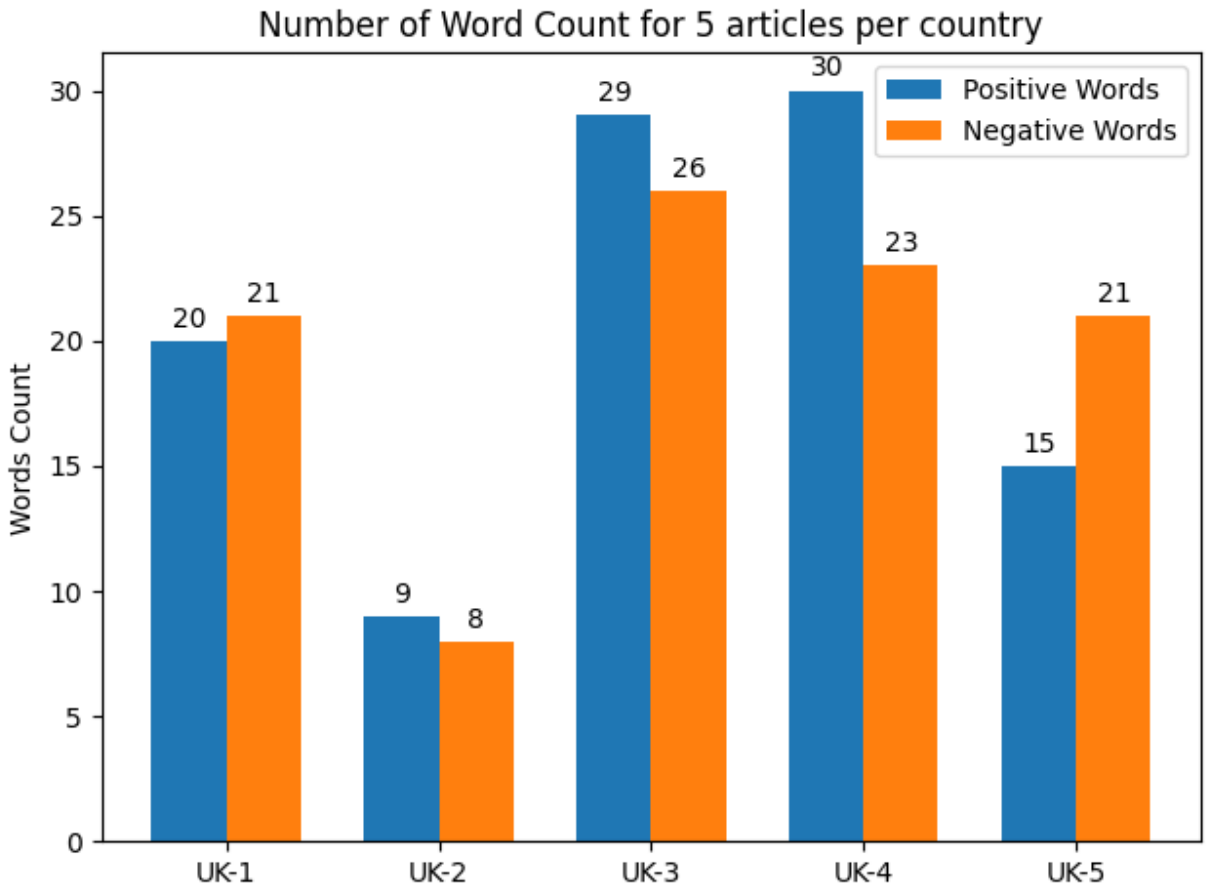


Figure 1.12: Total positive and negative word count for 5 articles of United Kingdom

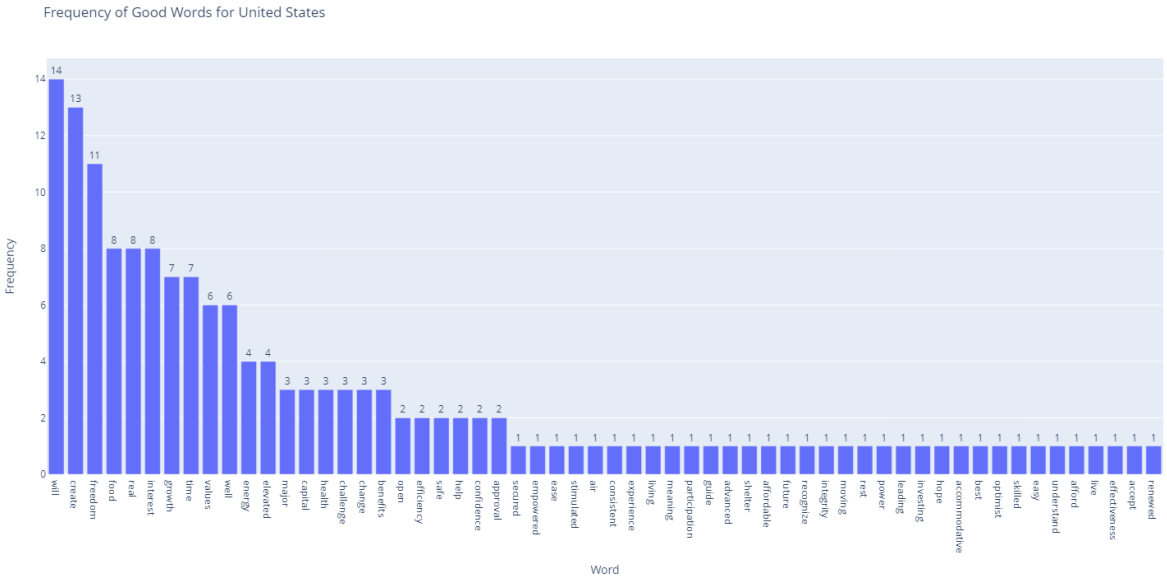


Figure 1.13: Total positive word count for 5 articles of United States

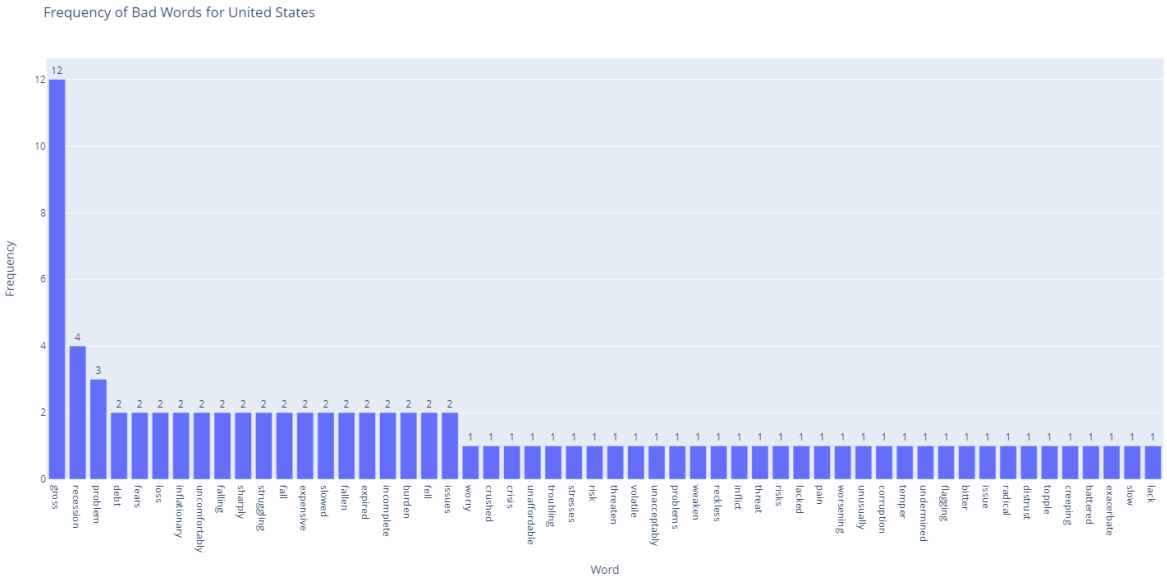


Figure 1.14: Total negative word count for 5 articles of United States

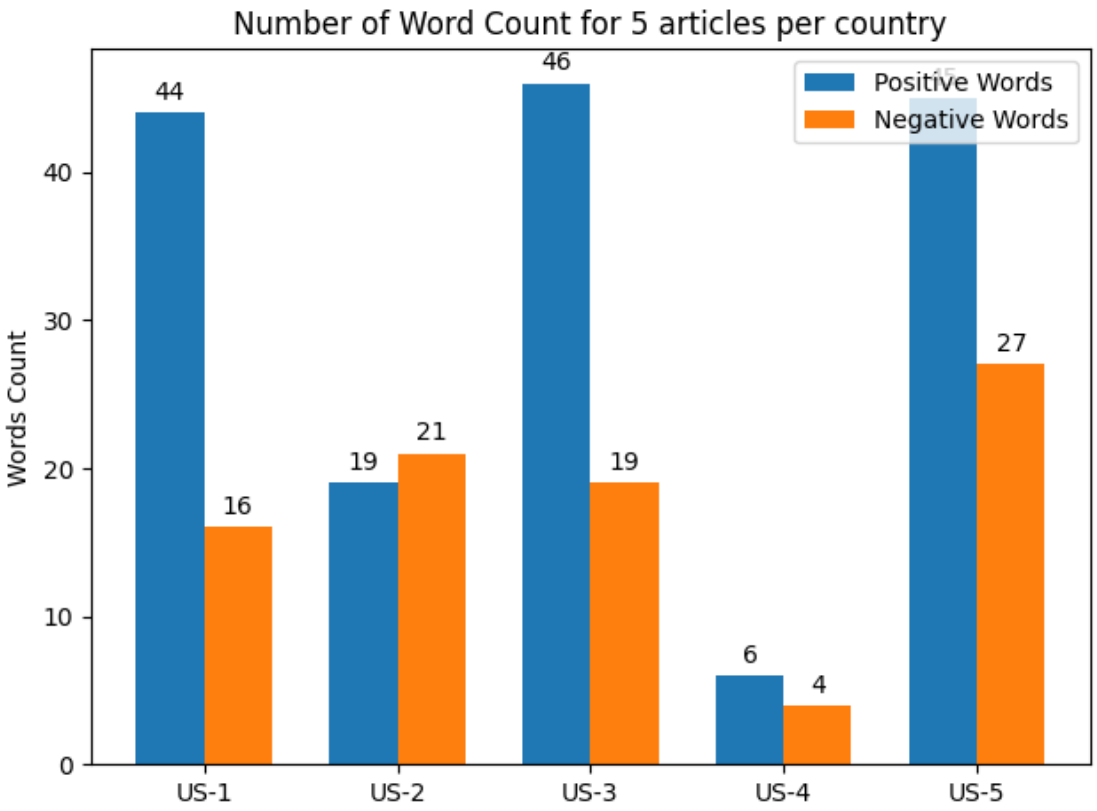


Figure 1.15: Total positive and negative word count for 5 articles of United States

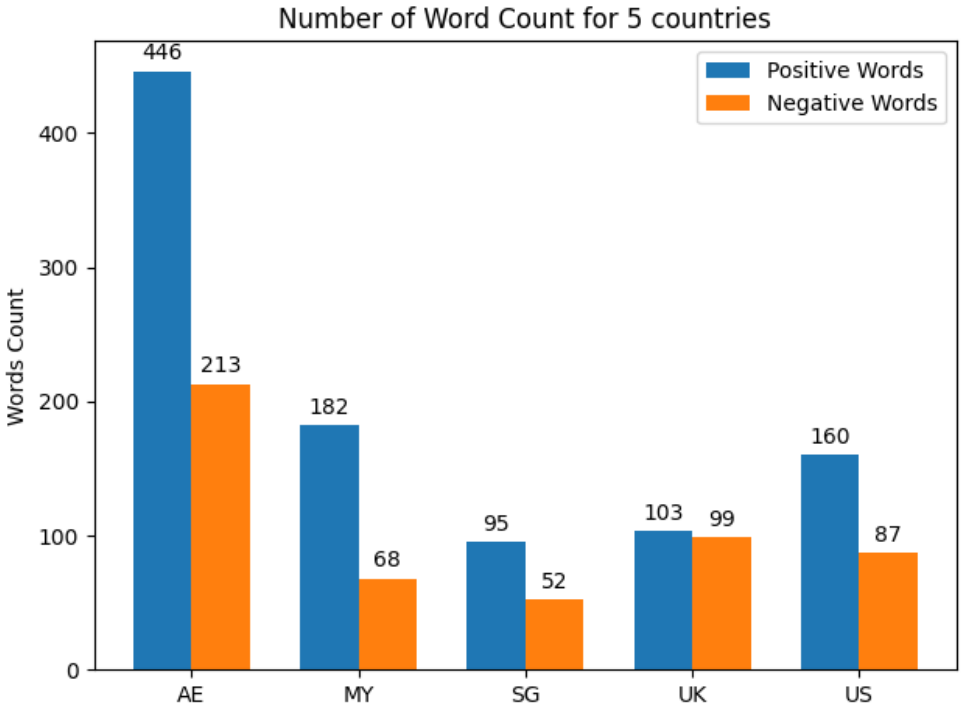


Figure 1.16: Overview of positive and negative word count for 5 countries

Problem 2

Problem Description

Moonbucks usually delivers stocks from a warehouse in the region. To optimize delivery, Moonbucks intends to have a local central distribution center in each country. Hence, we have to determine which store to use for the local distribution center with randomly selected 7 stores in the 5 countries respectively. For each store, we calculate the standard deviation of distance to all other stores and find the lowest value to set it as a distribution center. Then, by using A* search, we find the shortest path starting from and ending at the distribution center as an optimal delivery route for delivery trucks. We also keep track of the total distance of the optimal delivery for each country. We show the results including the pushpins maps and route maps in Bing Maps by using the Bing Maps Distance Matrix API.

Tools and Algorithms

API (Bing Map Distance Matrix API)

To activate this API, we must register an account in the Bing maps portal. After the account registration, we must create an API key by providing some information such as application name, key type, and application type. We only used the basic key in our group project.

1) Route-Distance Matrix

This API provides travel time and distances for a set of origins and destinations. The distance and times returned are based on the routes calculated by Bing Map Route API. HTTP GET Request URL is created to get a distance matrix between an origin and a set of destinations. The responses are shown in JSON formats.

Example HTTP Get Request URL:

https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=25.215973,55.409539&destinations=25.215973,55.409539;25.245717,55.359237;25.321535,55.375703;25.25374,55.30349

9;25.33381,55.375645;25.132344,55.118461;24.350423,54.569979&travelMode=driving&key={BingMapsKey}

2) RESTImagery-StaticMap & RESTImagery-StaticMapWithRoute

These APIs provide a function to display a route on a static map by requesting static map metadata. Static map metadata includes latitude, longitude, size of pushpins, map area and center point.

Example HTTP to get a map with Road imagery and declutter overlapping pushpins:

https://dev.virtualearth.net/REST/v1/Imagery/Map/Road?&pp=25.132344,55.118461;;C&pp=25.25374,55.303499;;1&pp=25.245717,55.359237;;2&pp=25.33381,55.375645;;3&pp=25.321535,55.375703;;4&pp=25.215973,55.409539;;5&pp=24.350423,54.569979;;6&pp=25.132344,55.118461;;7&dcl=1&mapSize=1000,750&key={BingMapsKey}

Example URL to get a map with Road imagery that displays a route:

https://dev.virtualearth.net/REST/v1/Imagery/Map/Road/Routes?wp.0=25.132344,55.118461;64;0&wp.1=25.25374,55.303499;66;1&wp.2=25.245717,55.359237;66;2&wp.3=25.33381,55.375645;66;3&wp.4=25.321535,55.375703;66;4&wp.5=25.215973,55.409539;66;5&wp.6=24.350423,54.569979;66;6&wp.7=25.132344,55.118461;66;7&dcl=1&mapSize=1000,750&optimize=distance&key={BingMapsKey}

Library

We use some packages such as requests, csv, pandas, numpy, random and exists to solve this problem.

We use the random module to get a list of 7 unique stores chosen from the population set containing all stores in a state by using random sampling. We also apply the csv module to read and write tabular data in CSV format. By using a writer object, it helps to convert required data by writing rows on file objects. We then apply pandas to load the CSV files into a DataFrame to access required data from specific columns or rows by using loc and filter functions. We also use

NumPy to convert DataFrame into 2D arrays and convert them to lists to get specific data like latitudes and longitudes of each store.

Besides, we apply the requests module to send GET requests to the Bing Maps API and extract data in json format to access the required information by parsing down the type JSON object. It is used to access the internal features such as longitude and latitude, destination index and travel duration and write the data in csv files, naming them as RouteCostFor_state.csv to generate all possible routes with distances calculated. Furthermore, we use the existing module to do checking for using the old data without creating a new csv file or not.

Algorithm Description

We implemented A* Search Algorithm to find the optimal delivery routes for each country. A* is using heuristic methods to achieve optimality and completeness and is a variant of the best-first algorithm. It calculates the cost, f(n) by using the below formula to travel to all of the neighboring nodes, and enters the node with the lowest value of f(n).

$$f(n) = g(n) + h(n)$$

g(n) being the distances of the start node (local distribution center) to other nodes (stores)

h(n) being a heuristic approximation of the node's value, storing distance between 2 stores

In A* search algorithm, the open_node is a list of nodes which have not been visited, while closed_node is a list of nodes which have been visited, indicating the optimal route. It will loop until the open_node gets empty and find the node with the lowest value of f(n) to append it to the closed_node. Finally, we get the optimal routes with the total distances for each country being shown in bing maps.

Time Complexity

n = number of countries

m = total number of branches in all chosen countries

p = number of chosen branches of a country

q = number of open node

Note : q< = p = 7, n = 5

For each country : **O(n)**

Find all branches of the country : **O(m)**

Choose 7 random branches : **O(p)**

Calculate the best distribution center : **$O(p^2)$**

Calculate all combinations of routes and costs for each route : **$O(p^2)$**

A* search using routes and costs information : **$O(pq^2)$**

Overall time complexity : $O(n(m+p^2+pq^2))$

Source Code

```
apikey = "ur own api"
centerindex = 0


def generateCSV(criteria):
    df = pd.read_csv('../raw_data.csv')
    correct_state = df['state'] == criteria
    state_df = df[correct_state]
    arr = state_df.filter(items=['latitude', 'longitude']).values
    filename = criteria + ".csv"
    file = open(filename, 'w', encoding='UTF8', newline='')
    writer = csv.writer(file)
    for x in arr:
        writer.writerow(x)
    file.close


def choose7RandomBranches(state):
    filename = "Route/" + state + ".csv"
    df = pd.read_csv(filename, header=None)
    arr = np.array(df)
    arr2 = arr.tolist()
    randomIndex = random.sample(range(0, len(arr)), 7)
    coordinates = []
    for x in randomIndex:
        coordinates.append(arr2[x])
    return coordinates


def generatingPossibleRoute(branches, state, useolddata):
    filename = "Route/RouteCostFor_" + state + ".csv"
    if useolddata & exists(filename):
        # read the csv and write into AllInfo, then return
        # if use old data, random branches must be manually defined as same as in the csv
        AllInfo = []
        df = pd.read_csv(filename)
        arr = np.array(df)
        arr2 = arr.tolist()
        for i in arr2:
            AllInfo.append(i)
        return AllInfo
    list_of_jsonURL = []
    destination = ""
    for y in range(len(branches)):
        destination += str(branches[y])[1:len(str(branches[y])) - 1].replace(" ", "")
        if y == len(branches) - 1:
            break
        destination += ";"
    # print(destination)
    url1 = "https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins="
    url2 = "&destinations="
    url3 = "&travelMode=driving&key="
```

```
    # Generate json URL
    for i in branches:
        temp2 = str(i)
        url = url1 + temp2[1:len(temp2) - 1].replace(" ", "") + url2 + destination + url3 + apikey
        print(url)
        list_of_jsonURL.append(url)
    # 1*7=7 pairs per http, total 7 loops hence 7 http, 7*7=49 pairs (including self-to-self because
troublesome to remove and replace back)
    # Store json URL
    outputList = []
    for i in list_of_jsonURL:
        output = requests.get(i).json()
        outputList.append(output)

    # Extract destination Index and travel distance from origin to destination
    # Store the info in AllInfo list
    AllInfo = []
    filename = "Route/RouteCostFor_" + state + ".csv"
    file = open(filename, 'w', encoding='UTF8', newline='')
    writer = csv.writer(file)
    writer.writerow(["Origin", "location1", "destination", "location2", "distance"])
    for k in range(len(outputList)):
        for j in range(len(branches)):
            info = [
                k, str(branches[k])[1:len(str(branches[k])) - 1].replace(" ", ""),
                outputList[k]["resourceSets"][0]["resources"][0]["results"][j]["destinationIndex"],
                str(branches[j])[1:len(str(branches[j])) - 1].replace(" ", ""),
                outputList[k]["resourceSets"][0]["resources"][0]["results"][j]["travelDistance"]
            ]
            writer.writerow(info)
            AllInfo.append(info)
    file.close
    return AllInfo


def distributionCenter(branches, state, useolddata):
    # Choose the lowest std dev as distribution center
    allRoutes = generatingPossibleRoute(branches, state, useolddata)
    distances = []
    for i in range(len(allRoutes)):
        distances.append(allRoutes[i][4])
    stdList = []
    for i in range(0, len(allRoutes), 7):
        stdList.append(numpy.std(distances[i:i + 7]))
    global centerindex
    centerindex = stdList.index(min(stdList))
    return branches[centerindex]


# gn=destination to center
def getGN(df):
    gn = []
    df = df.loc[(df['destination'] == centerindex) & (df['distance'] != 0)]
    gn.append(df['distance'].values)
    return gn[0]
```

```
# hn=distance between 2 points
# [[0 to 1, 0 to 2,0 to 3 ...],[1 to 0, 1 to 2, 1 to 3...],...]
def getHN(df):
    hn = []
    tempHn = df["distance"].values
    listing = []
    for i in tempHn:
        if i != 0:
            listing.append(i)
    temp = []
    count = 0
    for i in range(len(listing)):
        if count % 5 == 0 and count != 0:
            temp.append(listing[i])
            hn.append(temp)
            count = 0
            temp = []
        else:
            temp.append(listing[i])
            count += 1

    return hn


RouteForPlotting = []
totalDistances = []


def A_star_search(fn, open_node, closed_node, branches, df):
    # fn is a 2d array which recorded the costs as [[0 to 1, 0 to 2,0 to 3 ...],[1 to 0, 1 to 2, 1 to 3...],...]
    if not bool(open_node):
        totalDistance = 0
        closed_node.append(centerindex)
        print("The route is: " + str(closed_node))
        coordinate = ""
        ctr = 0
        for i in range(len(closed_node)):
            coordinate += str(branches[closed_node[i]])
            RouteForPlotting.append(branches[closed_node[i]])
            if ctr < len(closed_node) - 1:
                totalDistance += df['distance'].where(
                    (df['Origin'] == closed_node[i]) & (df['destination'] == closed_node[i + 1])).sum()
                coordinate += "->"
                ctr += 1
        print(coordinate)
        totalDistances.append(totalDistance)
        print("Total Distance = " + str(totalDistance) + " km")
        viewLocation = plotPushPins(RouteForPlotting)
        print("Click here to see the center:")
        print(viewLocation)
        viewMap = plotRouteInBingMap(RouteForPlotting)
        print("Click here to see the map:")
        print(viewMap + "\n")
```

```
        return
    if not bool(closed_node):
        open_node.remove(centerindex)
        closed_node.append(centerindex)
    still_open = False
    current_node = closed_node[len(closed_node) - 1]
    while not still_open:
        min_index = fn[current_node].index(min(fn[current_node]))
        actual_index = fn[current_node].index(min(fn[current_node]))
        if actual_index >= current_node:
            min_index += 1
        for i in open_node:
            if i == min_index:
                still_open = True
                break
        if not still_open:
            fn[current_node][actual_index] = 999999
    open_node.remove(min_index)
    closed_node.append(min_index)
    A_star_search(fn, open_node, closed_node, branches, df)


def getFN(h, g):
    fn = []
    allFN = []
    for i in h:
        for x in range(6):
            allFN.append(i[x] + g[x])

    temp = []
    count = 0
    for i in range(len(allFN)):
        if count % 5 == 0 and count != 0:
            temp.append(allFN[i])
            fn.append(temp)
            count = 0
            temp = []
        else:
            temp.append(allFN[i])
            count += 1

    return fn


# Plot pushpins only to show the location of "CENTER DISTRIBUTION"
def plotPushPins(location):
    starter = "&dcl=1&mapSize=1000,750&key="
    api_key = starter + "ur own api"
    URLstruture = "https://dev.virtualearth.net/REST/v1/Imagery/Map/Road?"
    URL = ""
    f1 = "&pp="
    f2 = ","
    f3 = ";;"
    f4 = "C"
    index = 0
```

```
    for i in location:
        if index == 0:
            URL = URL + f1 + str(i[0]) + f2 + str(i[1]) + f3 + f4
        else:
            URL = URL + f1 + str(i[0]) + f2 + str(i[1]) + f3 + str(index)
        index += 1
        if index == 8:
            break

    URL = URL + api_key
    link = URLstruture + URL
    return link


# To generete BING MAP showing the complete route from origin to different location
# Finally back to origin
def plotRouteInBingMap(RouteList):
    starter = "&dcl=1&mapSize=1000,750&optimize=distance&key="
    api_key = starter + "ur own api"
    URLstructure = "https://dev.virtualearth.net/REST/v1/Imagery/Map/Road/Routes?"
    URL = ""
    f1 = "wp."
    f2 = ","
    index = 0
    f3 = ";64;"
    f4 = "="
    f5 = ";66;"
    f6 = "&"
    for x in RouteList:
        if index == 0:
            URL = URL + f1 + str(index) + f4 + str(x[0]) + f2 + str(x[1]) + f3 + str(index)
        else:
            URL = URL + f6 + f1 + str(index) + f4 + str(x[0]) + f2 + str(x[1]) + f5 + str(index)
        index += 1
        if index == 8:
            URL = URL + api_key
            break
    link = URLstructure + URL
    RouteForPlotting.clear()
    return link

states_p2 = ["AE", "MY", "SG", "GB", "US"]
for x in states_p2:
    generateCSV(x)
    randomBranches = choose7RandomBranches(x)
    print(x + "'s Randomly Selected Branches: ")
    print(randomBranches)
    # plotPushPins(randomBranches)
    center = distributionCenter(randomBranches, x,
                                False)
    # If you want to use old data without api calls, change to true
    # For without api call, if the csv file does not exist, it will still proceed with api call and generate file
    print("Distribution center for " + x + " is " + str(center))
    print("Index of distribution center: " + str(centerindex))
    df = pd.read_csv('Route/RouteCostFor_' + x + '.csv')
```

```
h = getHN(df)
g = getGN(df)
# f(n) = g(n) + h(n)
f = getFN(h, g)
A_star_search(f, [0, 1, 2, 3, 4, 5, 6], [], randomBranches, df)
```

Snapshots of Input/Output

```
AE's Randomly Selected Branches:
[[25.215973, 55.409539], [25.245717, 55.359237], [25.321535, 55.375703], [25.25374, 55.303499], [25.33381, 55.375645], [25.132344, 55.118461], [24.350423, 54.569979]]
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=25.215973,55.409539&destinations=25.215973,55.409539;25.245717,55.359237;25.321535,55.375703;25.25374,55.303499
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=25.245717,55.359237&destinations=25.215973,55.409539;25.245717,55.359237;25.321535,55.375703;25.25374,55.303499
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=25.321535,55.375703&destinations=25.215973,55.409539;25.245717,55.359237;25.321535,55.375703;25.25374,55.303499
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=25.25374,55.303499&destinations=25.215973,55.409539;25.245717,55.359237;25.321535,55.375703;25.25374,55.303499;
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=25.33381,55.375645&destinations=25.215973,55.409539;25.245717,55.359237;25.321535,55.375703;25.25374,55.303499;
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=25.132344,55.118461&destinations=25.215973,55.409539;25.245717,55.359237;25.321535,55.375703;25.25374,55.303499
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=24.350423,54.569979&destinations=25.215973,55.409539;25.245717,55.359237;25.321535,55.375703;25.25374,55.303499
Distribution center for AE is [25.132344, 55.118461]
Index of distribution center: 5
The route is: [5, 3, 1, 4, 2, 0, 6, 5]
[25.132344, 55.118461]->[25.25374, 55.303499]->[25.245717, 55.359237]->[25.33381, 55.375645]->[25.321535, 55.375703]->[25.215973, 55.409539]->[24.350423, 54.569979]->[25.132344,
Total Distance = 356.658 km
Click here to see the center:
https://dev.virtualearth.net/REST/v1/Imagery/Map/Road?&pp=25.132344,55.118461;;C&pp=25.25374,55.303499;;1&pp=25.245717,55.359237;;2&pp=25.33381,55.375645;;3&pp=25.321535,55.37570
Click here to see the map:
https://dev.virtualearth.net/REST/v1/Imagery/Map/Road/Routes?wp.0=25.132344,55.118461;64;0&wp.1=25.25374,55.303499;66;1&wp.2=25.245717,55.359237;66;2&wp.3=25.33381,55.375645;66;3
```

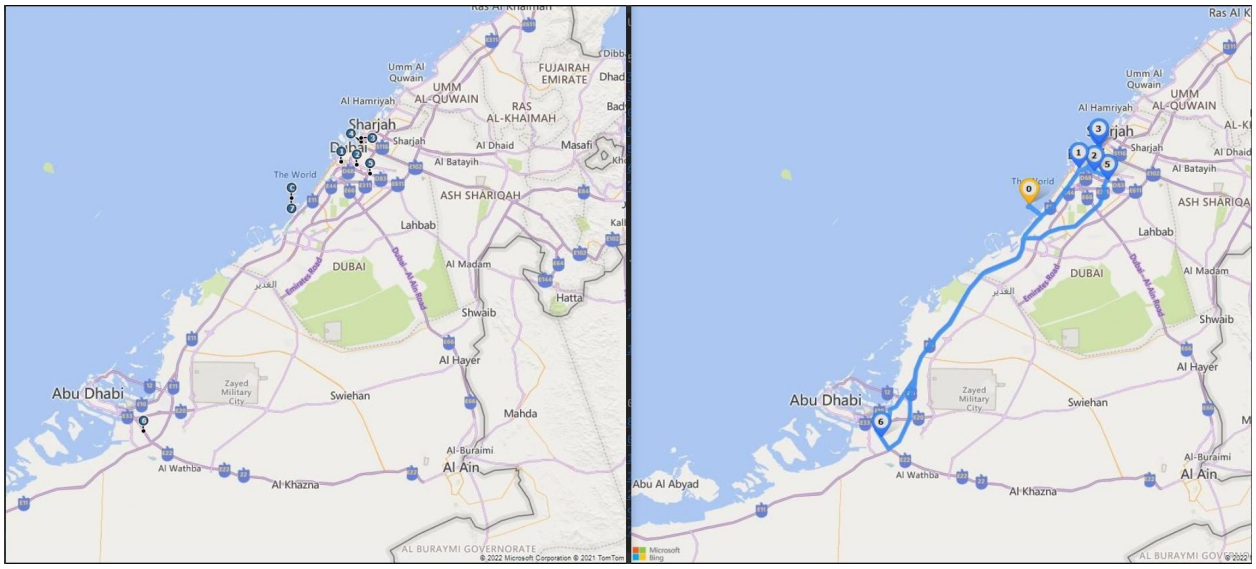


Figure 2.1: Map for United Arab Emirates (pushpins and routes)

```
MY's Randomly Selected Branches:
[[3.080939, 101.580989], [5.338058, 100.428507], [3.090996, 101.74241], [3.146109, 101.698017], [2.991867, 101.794626], [3.160322, 101.700699], [3.137839, 101.610273]]
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=3.080939,101.580989&destinations=3.080939,101.580989;5.338058,100.428507;3.090996,101.74241;3.146109,101.698017
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=5.338058,100.428507&destinations=3.080939,101.580989;5.338058,100.428507;3.090996,101.74241;3.146109,101.698017
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=3.090996,101.74241&destinations=3.080939,101.580989;5.338058,100.428507;3.090996,101.74241;3.146109,101.698017
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=3.146109,101.698017&destinations=3.080939,101.580989;5.338058,100.428507;3.090996,101.74241;3.146109,101.698017
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=2.991867,101.794626&destinations=3.080939,101.580989;5.338058,100.428507;3.090996,101.74241;3.146109,101.698017
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=3.160322,101.700699&destinations=3.080939,101.580989;5.338058,100.428507;3.090996,101.74241;3.146109,101.698017
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=3.137839,101.610273&destinations=3.080939,101.580989;5.338058,100.428507;3.090996,101.74241;3.146109,101.698017
Distribution center for MY is [3.137839, 101.610273]
Index of distribution center: 6
The route is: [6, 0, 3, 2, 4, 5, 1, 6]
[3.137839, 101.610273]->[3.080939, 101.580989]->[3.146109, 101.698017]->[3.090996, 101.74241]->[2.991867, 101.794626]->[3.160322, 101.700699]->[5.338058, 100.428507]->[3.137839,
Total Distance = 739.394 km
Click here to see the center:
https://dev.virtualearth.net/REST/v1/Imagery/Map/Road?&pp=3.137839,101.610273;;C&pp=3.080939,101.580989;;1&pp=3.146109,101.698017;;2&pp=3.090996,101.74241;;3&pp=2.991867,101.7946
Click here to see the map:
https://dev.virtualearth.net/REST/v1/Imagery/Map/Road/Routes?wp.0=3.137839,101.610273;64;0&wp.1=3.080939,101.580989;66;1&wp.2=3.146109,101.698017;66;2&wp.3=3.090996,101.74241;66;
```

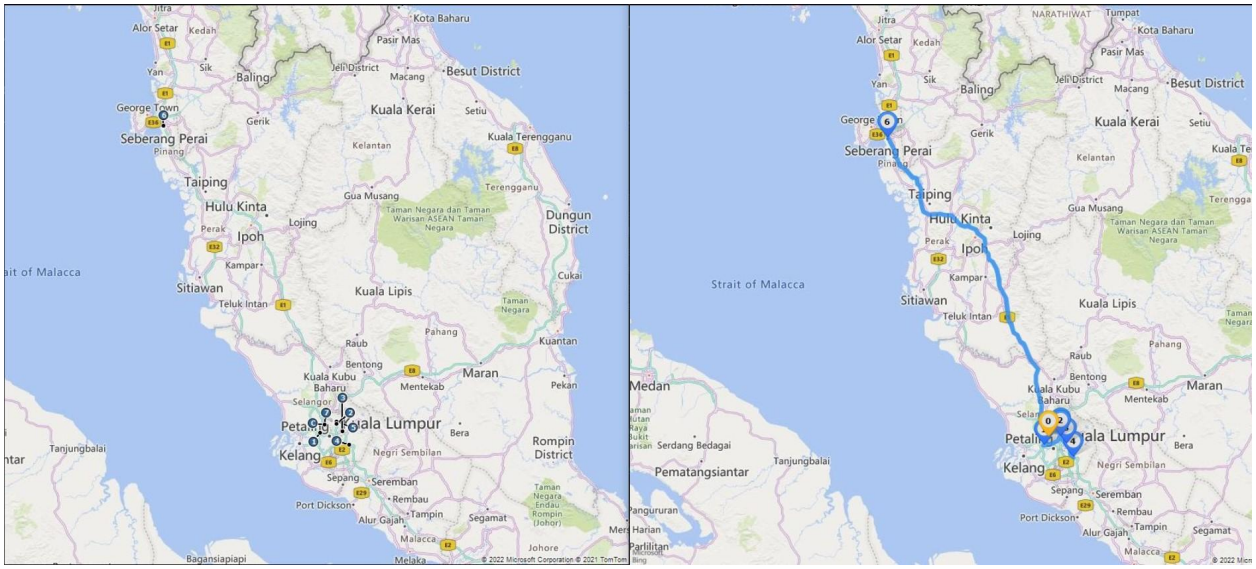


Figure 2.2: Map for Malaysia (pushpins and routes)

```
SG's Randomly Selected Branches:
[[1.317127, 103.84391], [1.291543, 103.845116], [1.277485, 103.847437], [1.28441, 103.852087], [1.303621, 103.765456], [1.263387, 103.821049], [1.312964, 103.925447]]
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=1.317127,103.84391&destinations=1.317127,103.84391;1.291543,103.845116;1.277485,103.847437;1.28441,103.852087;1
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=1.291543,103.845116&destinations=1.317127,103.84391;1.291543,103.845116;1.277485,103.847437;1.28441,103.852087;
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=1.277485,103.847437&destinations=1.317127,103.84391;1.291543,103.845116;1.277485,103.847437;1.28441,103.852087;
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=1.28441,103.852087&destinations=1.317127,103.84391;1.291543,103.845116;1.277485,103.847437;1.28441,103.852087;1
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=1.303621,103.765456&destinations=1.317127,103.84391;1.291543,103.845116;1.277485,103.847437;1.28441,103.852087;
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=1.263387,103.821049&destinations=1.317127,103.84391;1.291543,103.845116;1.277485,103.847437;1.28441,103.852087;
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=1.312964,103.925447&destinations=1.317127,103.84391;1.291543,103.845116;1.277485,103.847437;1.28441,103.852087;
Distribution center for SG is [1.291543, 103.845116]
Index of distribution center: 1
The route is: [1, 3, 2, 5, 0, 6, 4, 1]
[1.291543, 103.845116]->[1.28441, 103.852087]->[1.277485, 103.847437]->[1.263387, 103.821049]->[1.317127, 103.84391]->[1.312964, 103.925447]->[1.303621, 103.765456]->[1.291543, 1
Total Distance = 70.543 km
Click here to see the center:
https://dev.virtualearth.net/REST/v1/Imagery/Map/Road?&pp=1.291543,103.845116;;C&pp=1.28441,103.852087;;1&pp=1.277485,103.847437;;2&pp=1.263387,103.821049;;3&pp=1.317127,103.8439
Click here to see the map:
https://dev.virtualearth.net/REST/v1/Imagery/Map/Road/Routes?wp.0=1.291543,103.845116;64;0&wp.1=1.28441,103.852087;66;1&wp.2=1.277485,103.847437;66;2&wp.3=1.263387,103.821049;66;
```

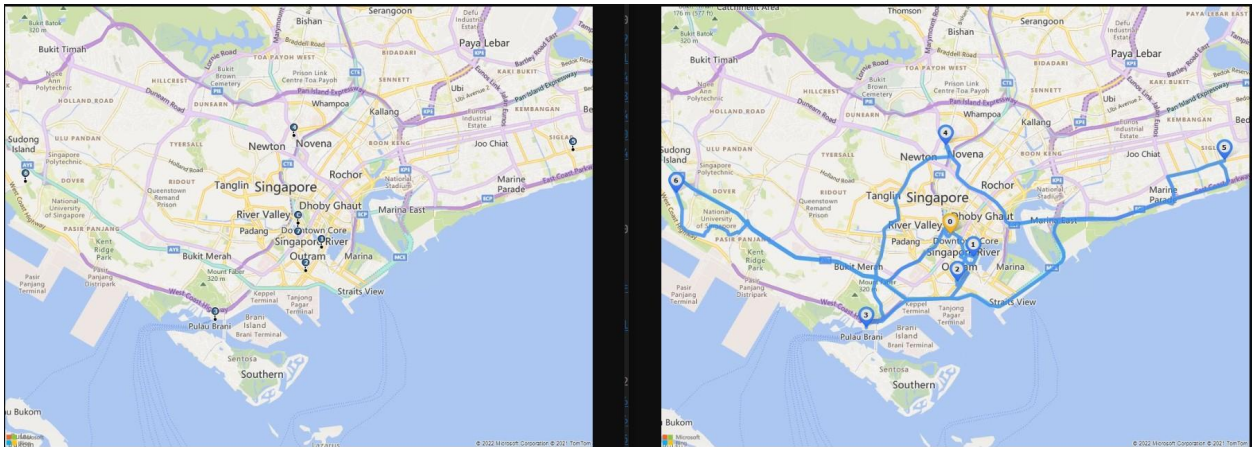


Figure 2.3: Map for Singapore (pushpins and routes)

```
GB's Randomly Selected Branches:
[[54.956884, -1.668226], [53.745313, -0.346994], [53.466772, -2.343539], [51.507114, -0.072531], [51.53015, -0.18612], [51.73798, -1.09713], [51.716061, -0.282385]]
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=54.956884,-1.668226&destinations=54.956884,-1.668226;53.745313,-0.346994;53.466772,-2.343539;51.507114,-0.07253
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=53.745313,-0.346994&destinations=54.956884,-1.668226;53.745313,-0.346994;53.466772,-2.343539;51.507114,-0.07253
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=53.466772,-2.343539&destinations=54.956884,-1.668226;53.745313,-0.346994;53.466772,-2.343539;51.507114,-0.07253
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=51.507114,-0.072531&destinations=54.956884,-1.668226;53.745313,-0.346994;53.466772,-2.343539;51.507114,-0.07253
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=51.53015,-0.18612&destinations=54.956884,-1.668226;53.745313,-0.346994;53.466772,-2.343539;51.507114,-0.072531;
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=51.73798,-1.09713&destinations=54.956884,-1.668226;53.745313,-0.346994;53.466772,-2.343539;51.507114,-0.072531;
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=51.716061,-0.282385&destinations=54.956884,-1.668226;53.745313,-0.346994;53.466772,-2.343539;51.507114,-0.07253
Distribution center for GB is [53.466772, -2.343539]
Index of distribution center: 2
The route is: [2, 1, 0, 5, 4, 6, 3, 2]
[53.466772, -2.343539]->[53.745313, -0.346994]->[54.956884, -1.668226]->[51.73798, -1.09713]->[51.53015, -0.18612]->[51.716061, -0.282385]->[51.507114, -0.072531]->[53.466772, -2
Total Distance = 1326.8799999999999 km
Click here to see the center:
https://dev.virtualearth.net/REST/v1/Imagery/Map/Road?&pp=53.466772,-2.343539;;C&pp=53.745313,-0.346994;;1&pp=54.956884,-1.668226;;2&pp=51.73798,-1.09713;;3&pp=51.53015,-0.18612;
Click here to see the map:
https://dev.virtualearth.net/REST/v1/Imagery/Map/Road/Routes?wp.0=53.466772,-2.343539;64;0&wp.1=53.745313,-0.346994;66;1&wp.2=54.956884,-1.668226;66;2&wp.3=51.73798,-1.09713;66;3
```

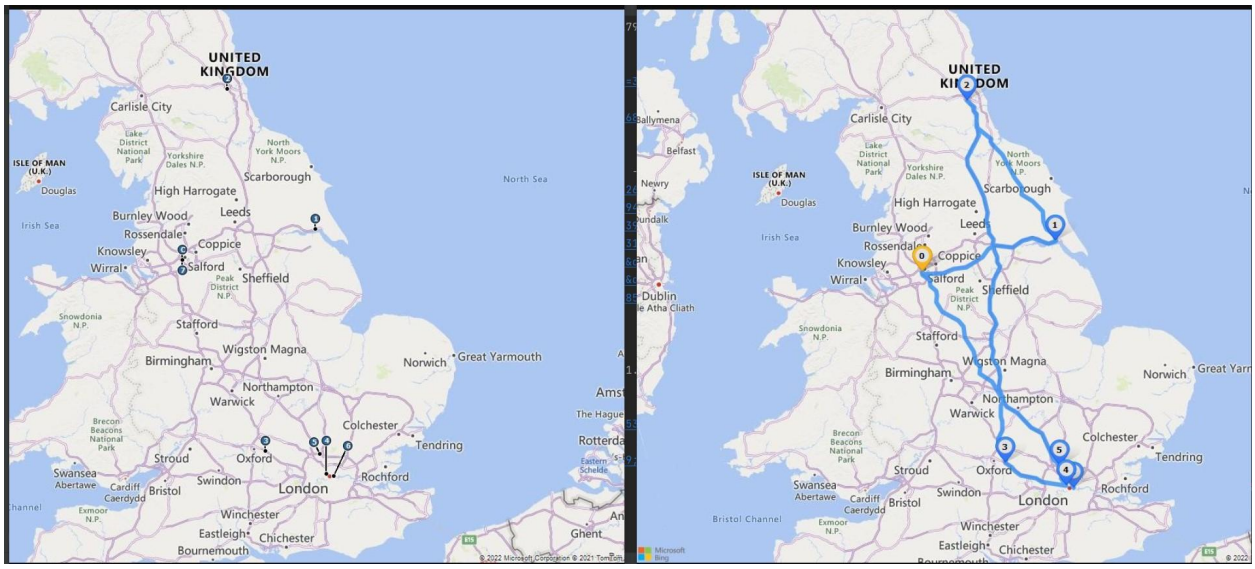


Figure 2.4: Map for United Kingdom (pushpins and routes)

```
US's Randomly Selected Branches:
[[37.351794, -121.967752], [40.764395, -73.726204], [42.927999, -72.305493], [32.862338, -96.854268], [32.841864, -116.981137], [33.832005, -118.029778], [38.427646, -122.741065]]
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=37.351794,-121.967752&destinations=37.351794,-121.967752;40.764395,-73.726204;42.927999,-72.305493;32.862338,-9
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=40.764395,-73.726204&destinations=37.351794,-121.967752;40.764395,-73.726204;42.927999,-72.305493;32.862338,-96
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=42.927999,-72.305493&destinations=37.351794,-121.967752;40.764395,-73.726204;42.927999,-72.305493;32.862338,-96
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=32.862338,-96.854268&destinations=37.351794,-121.967752;40.764395,-73.726204;42.927999,-72.305493;32.862338,-96
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=32.841864,-116.981137&destinations=37.351794,-121.967752;40.764395,-73.726204;42.927999,-72.305493;32.862338,-9
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=33.832005,-118.029778&destinations=37.351794,-121.967752;40.764395,-73.726204;42.927999,-72.305493;32.862338,-9
https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=38.427646,-122.741065&destinations=37.351794,-121.967752;40.764395,-73.726204;42.927999,-72.305493;32.862338,-9
Distribution center for US is [32.862338, -96.854268]
Index of distribution center: 3
The route is: [3, 4, 5, 0, 6, 1, 2, 3]
[32.862338, -96.854268]->[32.841864, -116.981137]->[33.832005, -118.029778]->[37.351794, -121.967752]->[38.427646, -122.741065]->[40.764395, -73.726204]->[42.927999, -72.305493]-
Total Distance = 10986.241 km
Click here to see the center:
https://dev.virtualearth.net/REST/v1/Imagery/Map/Road?&pp=32.862338,-96.854268;;C&pp=32.841864,-116.981137;;1&pp=33.832005,-118.029778;;2&pp=37.351794,-121.967752;;3&pp=38.427646
Click here to see the map:
https://dev.virtualearth.net/REST/v1/Imagery/Map/Road/Routes?wp.0=32.862338,-96.854268;64;0&wp.1=32.841864,-116.981137;66;1&wp.2=33.832005,-118.029778;66;2&wp.3=37.351794,-121.96
```

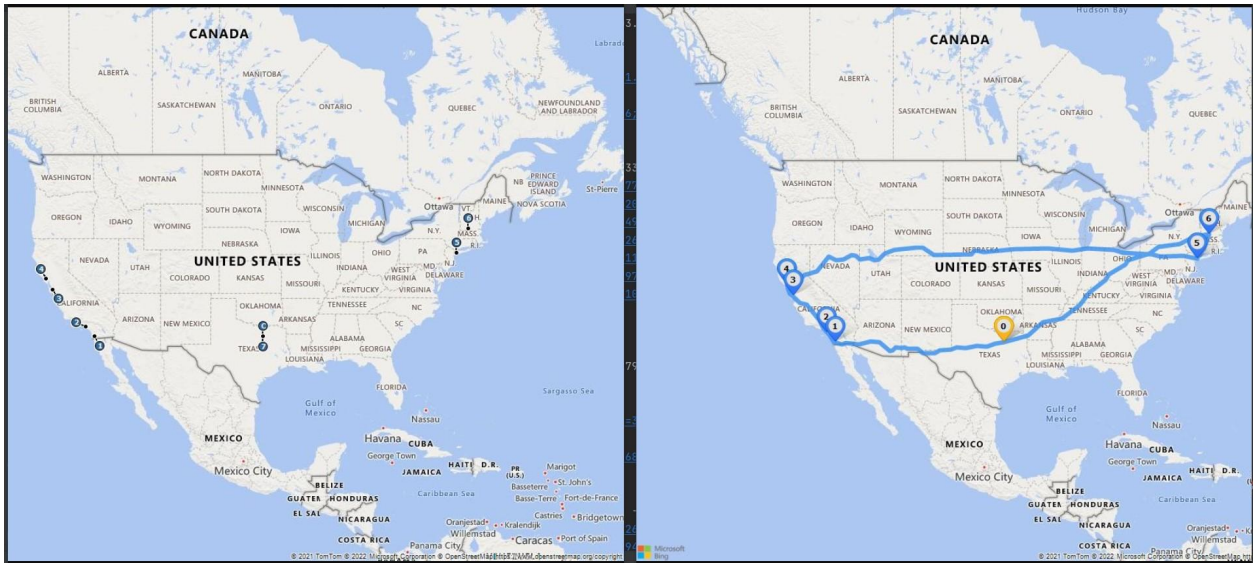


Figure 2.5: Map for United States of America (pushpins and routes)

Problem 3

Problem Description

To successfully expand the business in the selected country, Moonbucks must also consider the running cost for delivering logistics instead of just focusing on the local economic and social situation of the country. In this problem, our group is asked to analyze the most suitable country that has a good local economic and social situation with the lowest optimal delivery. Then, we will determine the final ranking of countries where new stores can be located. To tackle this problem, we first initialize ‘scoreA’ = word ratio of good words/total words where the data can be retrieved from problem 1, and initialize ‘scoreB’ = distance ratio using the formula (1/(distance_list/sum(distance_list)) where the data can be retrieved from problem 2, and perform normalization to produce an accurate data. Lastly, we use formula ((scoreA * scoreB) * 100) and choose the country with the highest percentage as the most suitable location to expand business.

Tools and Algorithms

Library

The sklearn.preprocessing package benefits from standardization of the data set especially when some outliers exist in the set. We apply the function normalize as it is quicker and easier to scale individual samples to have unit norm, finally get the distance ratios for each country.

Time Complexity

Word ratio: **O(n)**

Distance ratio: **O(n)**

Final score: **O(n)**

Total time complexity = O(3n) = **O(n)**

Source Code

```
# Result from Problem 1 (Assumption only)
df_for_word = pd.read_csv('csv/Overview.csv')
good_words = df_for_word['positive_word'].tolist()
bad_words = df_for_word['negative_word'].tolist()

# Result from Problem 2
# distance_list = [356.658, 739.394, 70.543, 1326.8799, 10986.241] # Sample
distance_list = totalDistances


def wordRatio(goodwords, badwords):
    country = ["AE", "MY", "SG", "UK", "US"]
    problist = []
    for i in range(len(country)):
        ratio = (goodwords[i]) / (goodwords[i] + badwords[i])
        problist.append(ratio)
    return problist


def distanceRatio():
    country = ["AE", "MY", "SG", "UK", "US"]
    distance_ratio = []
    for i in range(len(country)):
        distance_ratio.append(1 / (distance_list[i] / sum(distance_list)))
    normalized = preprocessing.normalize([distance_ratio])
    return normalized[0]


def calculateScore(score1, score2):
    country = ["AE", "MY", "SG", "UK", "US"]
    score = []
    for i in range(len(country)):
        score.append(round(((score1[i] * score2[i]) * 100), 5))
    return score
```

```
country_p3 = ["AE", "MY", "SG", "GB", "US"]
print("\nProblem 3")
scoreA = wordRatio(good_words, bad_words)
print("Word Score For Each Country:")
for i in range(len(country_p3)):
   print(str(country_p3[i]) + ": " + str(scoreA[i]))

scoreB = distanceRatio()
print("\nDistance Score For Each Country:")
for i in range(len(country_p3)):
   print(str(country_p3[i]) + ": " + str(scoreB[i]))

finalScore = calculateScore(scoreA, scoreB)
print("\nScore for all country based on the local economic and lowest optimal delivery:")
for i in range(len(country_p3)):
   print(str(country_p3[i]) + ": " + str(finalScore[i]) + " %")

print("\nThe most recommended countries based on optimal distance and positive sentiment is",
    country_p3[finalScore.index(max(finalScore))] + ".")
```

SnapShot Input/Output

```
Problem 3
Word Score For Each Country:
AE: 0.6767830045523521
MY: 0.728
SG: 0.6462585034013606
GB: 0.5099009900990099
US: 0.6477732793522267

Distance Score For Each Country:
AE: 0.19292205862218173
MY: 0.09305890443264361
SG: 0.9753936688838027
GB: 0.0518563854830193
US: 0.006263033514745407

Score for all country based on the local economic and lowest optimal delivery:
AE: 13.05664 %
MY: 6.77469 %
SG: 63.03565 %
GB: 2.64416 %
US: 0.4057 %

The most recommended countries based on optimal distance and positive sentiment is SG.
```

Figure 3.1: Score for 5 counties and recommendation

Conclusion

First and foremost, based on the result we get from problem 1 using Brute Force Algorithms, we decided that Malaysia is the best country for Moonbucks to expand their business because Malaysia has the highest rate of positive words in sentiment analysis of the country's local economy and social situation.

In problem 2, we determined the store to be used for the local distribution center by randomly selecting 7 stores in the 5 countries respectively. For each store, we set the store with lowest standard deviation as the distribution center. Our group was able to find the shortest path starting from and ending at the distribution center as an optimal delivery route for delivery trucks. By using Bing Map API, an API key is obtained in order to calculate the distances and plot the most optimal route for each of the delivery trucks.

In problem 3, we developed a formula for calculating the overall score in percentage for each of the five countries, then we selected the country with the greatest score as the next location for Moonbucks to establish a branch. The formula is (score A * score B) * 100, where score A is the good words ratio, and score B is the shortest distance ratio. As Singapore has the highest percentage, we can conclude that Singapore is the most favorable place for business expansion.

In conclusion, our team is able to do analysis based on the dataset that contains the location of Moonbucks stores all over the world. The results of the analysis can provide insights to Moonbucks management team in making business decisions by considering the local economics or social situation of the country and the running cost for delivery logistics.

References

Aravindan, A. (2022, February 17). *Singapore keeps 2022 growth forecast as economy stays on recovery path*. Reuters. Retrieved June 7, 2022, from https://www.reuters.com/markets/asia/singapore-slightly-upgrades-q4-gdp-keeps-2022-forecast-2022-02-17/

Bar. (n.d.). Plotly. Retrieved June 12, 2022, from https://plotly.com/python/bar-charts/

Bernama. (2021, December 31). *2022 will see Malaysia back on track of long term economic plan*. The Sun Daily. Retrieved June 6, 2022, from https://www.thesundaily.my/local/2022-will-see-malaysia-back-on-track-of-long-term-economic-plan-LC8712793

Economic and political overview in the United Arab Emirates. (2022). Crédit Agricole Group. Retrieved June 6, 2022, from https://international.groupecreditagricole.com/en/international-support/united-arab-emirates/economic-overview

Elessawy, F. M., Alshehhi, A. S., Alnaqbi, A. M., Dhaheri, A. H. A., Arafati, K. K. A., & Suwaidi, M. M. A. (2022). The Impact of COVID-19 on Some Socio-Economic Sectors in the United Arab Emirates. *Open Journal of Social Sciences*, *10*(04), 284–302. https://doi.org/10.4236/jss.2022.104021

Gross Domestic Product, First Quarter 2022 (Advance Estimate) | U.S. Bureau of Economic Analysis (BEA). (2022, April 28). Bureau of Economic Analysis. Retrieved June 7, 2022, from

https://www.bea.gov/news/2022/gross-domestic-product-first-quarter-2022-advance-estimate#:%7E:text=Real%20gross%20domestic%20product%20(GDP,real%20GDP%20increased%206.9%20percent.

Iwamoto, K. (2021, November 24). *Singapore expects 3–5% growth in 2022 amid "uneven" COVID rebound*. Nikkei Asia. Retrieved June 7, 2022, from https://asia.nikkei.com/Economy/Singapore-expects-3-5-growth-in-2022-amid-uneven-COVID-rebound

Jun, S. W. (2022, January 4). *Socio-Economic Research Centre: A better year in 2022 for Malaysian economy*. Malay Mail. Retrieved June 6, 2022, from https://www.malaymail.com/news/malaysia/2022/01/04/socio-economic-research-centre-a-better-year-in-2022-for-malaysian-economy/2032924

Lee, E. (2021, October 29). *Malaysian economy to grow faster at 5.5%-6.5% in 2022 vs 3%-4% in 2021*. The Edge Markets. Retrieved June 6, 2022, from https://www.theedgemarkets.com/article/malaysian-economy-grow-faster-5565-2022-vs-34-2021

Milliken, D., & Schomberg, W. (2022, May 12). *UK economy shrinks in March as recession risks mount*. Reuters. Retrieved June 7, 2022, from https://www.reuters.com/world/uk/uk-economy-shrinks-march-grows-08-q1-2022-05-12/

Negative Words That Start With A to Z: 4600 Sad Words List. (2015). Positive Words Research. https://positivewordsresearch.com/list-of-negative-words/

Reuters. (2022, May 12). *UK economy unexpectedly shrinks on recession risks*. Free Malaysia Today. Retrieved June 7, 2022, from https://www.freemalaysiatoday.com/category/business/2022/05/12/uk-economy-unexpectedly-shrinks-on-recession-risks/

Rushe, D. (2022, May 11). *US inflation rate slows but remains close to 40-year high*. The Guardian. Retrieved June 7, 2022, from https://www.theguardian.com/business/2022/may/11/us-inflation-consumer-price-index-figures

Smialek, J. (2022, May 12). *U.S. Inflation Is Still Climbing Rapidly*. The New York Times. Retrieved June 7, 2022, from https://www.nytimes.com/2022/05/11/business/economy/april-2022-cpi.html

Smith, E. (2022, May 12). *UK economy "only going to get worse" as growth slowdown begins*. CNBC. Retrieved June 7, 2022, from https://www.cnbc.com/2022/05/12/uk-economy-only-going-to-get-worse-as-growth-slowdown-begins.html

Subin, S., & McKeever, V. (2022, May 13). *U.S. Treasury prices slip as investors pile back into stocks*. CNBC. Retrieved June 7, 2022, from https://www.cnbc.com/2022/05/13/us-bonds-treasury-prices-slip-as-investors-pile-back-into-stocks.html

Technical Department of The Malaysian Institute of Certified Public Accountants (Micpa). (2022, April 4). *Malaysia economic outlook 2022 – firmer recovery despite headwinds*.

The Sun Daily. Retrieved June 6, 2022, from https://www.thesundaily.my/home/malaysia-economic-outlook-2022-firmer-recovery-despite-headwinds-AY9032763

Turkel, W. J., & Crymble, A. (2012, July 17). *Counting Word Frequencies with Python*. Programming Historian. Retrieved from https://doi.org/10.46430/phen0003

UAE To Post Strong Economic Growth In 2022 And 2023. (2022, February 24). Fitch Solutions. Retrieved June 6, 2022, from https://www.fitchsolutions.com/country-risk/uae-post-strong-economic-growth-2022-and-2023-24-02-2022?fSWebArticleValidation=true&mkt_tok=NzMyLUNLSC03NjcAAAGEV3RsDOE4Ru4vdXD8x_rylSC4XXsfv9bqGS6OUwx1bACGqaYvkt5yjnjKR8K7u5B7EQp9RA6yqRY7sYSKrJbaRW-_TYoW1DMDunaRWID41q7rgcRheQ

UK economy latest - Office for National Statistics. (2021, January 25). Office for National Statistics. Retrieved June 6, 2022, from https://www.ons.gov.uk/economy/economicoutputandproductivity/output/articles/ukeconomylatest/2021-01-25

UK economy shrinks in March, grows 0.8% in first quarter. (2022, May 12). CNBC. Retrieved June 6, 2022, from https://www.cnbc.com/2022/05/12/uk-economy-shrinks-in-march-grows-0point8percent-in-q1.html

United Arab Emirates. (2022, February). Coface. Retrieved June 6, 2022, from https://www.coface.com/Economic-Studies-and-Country-Risks/United-Arab-Emirates

United Arab Emirates Economy: Population, GDP, Inflation, Business, Trade, FDI, Corruption. (n.d.). The Heritage Foundation. Retrieved June 6, 2022, from https://www.heritage.org/index/country/unitedarabemirates

United States Economy: Population, GDP, Unemployment, Inflation, Spending. (2022). The Heritage Foundation. Retrieved June 7, 2022, from https://www.heritage.org/index/country/unitedstates

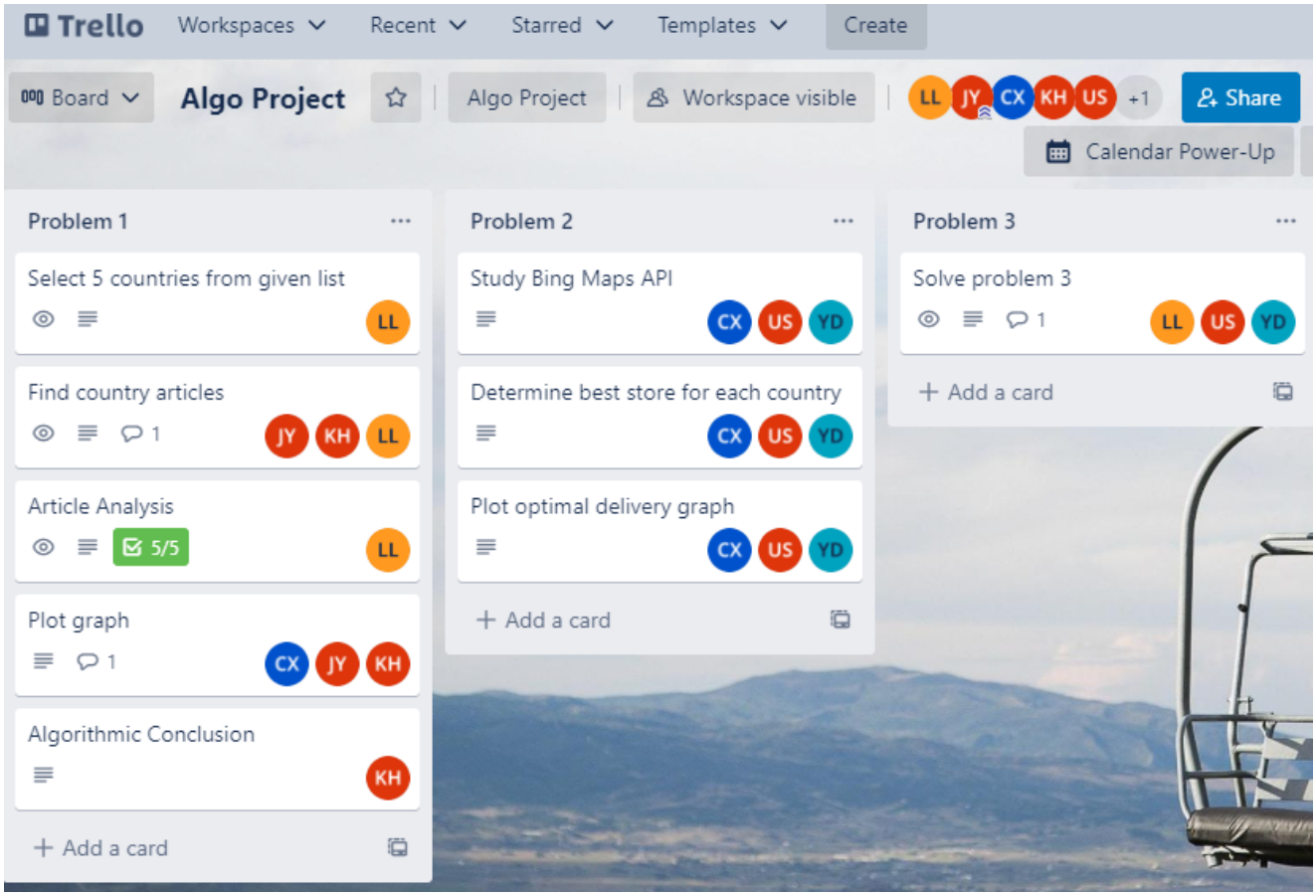
1400+ Positive Words List, Good Words, Nice Words A to Z. (n.d.). Positive Words Research. https://positivewordsresearch.com/list-of-positive-words/

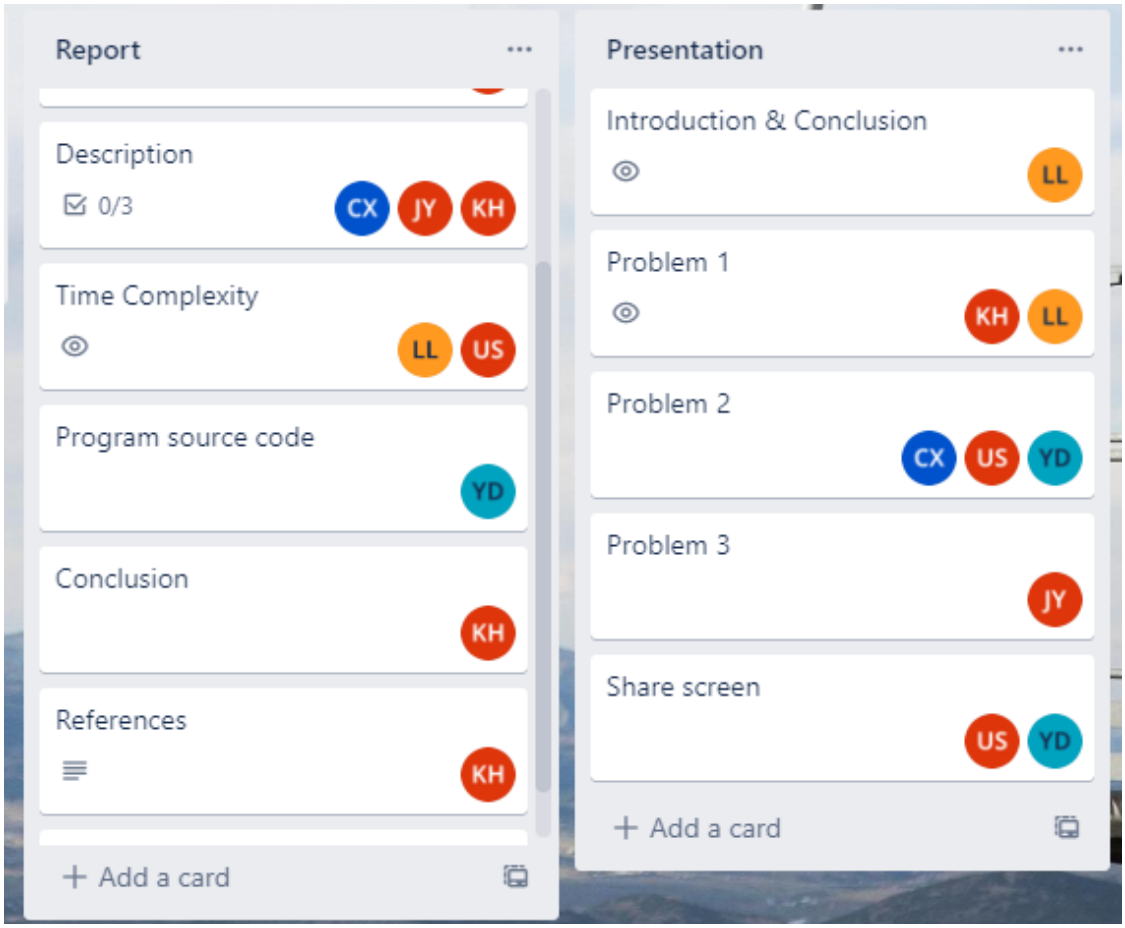
Appendices

Job delegation through trello:

https://trello.com/invite/b/SwcC0Pyj/e3a2c063b4cda86daa33b0c93c2ad218/algo-project

Screenshots:




Report
Description
0/3
CX
JY
KH
Time Complexity
LL
US
Program source code
YD
Conclusion
KH
References
KH
Add a card
Presentation
Introduction & Conclusion
LL
Problem 1
KH
LL
Problem 2
CX
US
YD
Problem 3
JY
Share screen
US
YD
Add a card