

In [1]:

```
import pandas, os, geopy, urllib.request, googlemaps
import csv
from geopy.distance import geodesic, lonlat, distance
from geopy.geocoders import Nominatim
import gmaps
gmaps.configure(api_key='AIzaSyAFG0b7rtDGzdV0anCTTpOYO_ZMLjNycqs')
#Problem 2
import matplotlib.pyplot as plt; plt.rcParamsdefaults()
import numpy as np
```

In [2]:

```
nom = geopy.geocoders.Nominatim(user_agent="my appl")
```

In [3]:

```
df1 = pandas.read_csv("Locations.csv")
df1["Coordinates"] = df1["Location"].apply(nom.geocode, timeout = 10)
df1["Latitude"] = df1["Coordinates"].apply(lambda x: x.latitude if x != None else None)
df1["Longitude"] = df1["Coordinates"].apply(lambda x: x.longitude if x != None else None)
)
```

In [4]:

```
import sys

class Vertex:
    def __init__(self, node):
        self.id = node
        self.adjacent = {}
        self.transport={}
        # Set distance to infinity for all nodes
        self.distance = sys.maxsize
        # Mark all nodes unvisited
        self.visited = False
        # Predecessor
        self.previous = None
        self.trans = None
    def __lt__(self, other):
        return self.distance < other.distance

    def get_connections(self):
        return self.adjacent.keys()

    def get_id(self):
        return self.id

    def get_weight(self, neighbor):
        return self.adjacent[neighbor]

    def add_neighbor(self, neighbor, trans, weight=0):
        self.adjacent[neighbor] = weight
        self.transport[neighbor] = trans

    def set_distance(self, dist):
        self.distance = dist

    def get_distance(self):
        return self.distance

    def set_previous(self, prev):
        self.previous = prev

    def set_visited(self):
        self.visited = True

    def __str__(self):
        return str(self.id) + ' adjacent: ' + str([x.id for x in self.adjacent])

class Graph:
    def __init__(self):
        self.vert_dict = {}
        self.num_vertices = 0

    def __iter__(self):
        return iter(self.vert_dict.values())

    def add_vertex(self, node):
        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(node)
        self.vert_dict[node] = new_vertex
        return new_vertex
```

```

def get_vertex(self, n):
    if n in self.vert_dict:
        return self.vert_dict[n]
    else:
        return None

def add_edge(self, frm, to, trans, cost = 0):
    if frm not in self.vert_dict:
        self.add_vertex(frm)
    if to not in self.vert_dict:
        self.add_vertex(to)

    self.vert_dict[frm].add_neighbor(self.vert_dict[to], trans, cost)

def get_vertices(self):
    return self.vert_dict.keys()

def set_previous(self, current):
    self.previous = current

def get_previous(self, current):
    return self.previous

def shortest(v, path, coster, transport_type):
    ''' make shortest path from v.previous'''
    if v.previous:
        path.append(v.previous.get_id())
        coster.append(v.previous.get_distance())
        shortest(v.previous, path, coster, transport_type)
    return

def dijkstra(aGraph, start):
    print ('''Dijkstra's shortest path''')
    # Set the distance for the start node to zero
    start.set_distance(0)

    # Put tuple pair into the priority queue
    unvisited_queue = [(v.get_distance(), v) for v in aGraph]
    heapq.heapify(unvisited_queue)

    while len(unvisited_queue):
        # Pops a vertex with the smallest distance
        uv = heapq.heappop(unvisited_queue)
        current = uv[1]
        current.set_visited()

        #for next in v.adjacent:
        for next in current.adjacent:
            # if visited, skip
            if next.visited:
                continue
            new_dist = current.get_distance() + current.get_weight(next)

            if new_dist < next.get_distance():
                next.set_distance(new_dist)
                next.set_previous(current)

        # Rebuild heap
        # 1. Pop every item

```

```

    while len(unvisited_queue):
        heapq.heappop(unvisited_queue)
        # 2. Put all vertices not visited into the queue
        unvisited_queue = [(v.get_distance(),v) for v in aGraph if not v.visited]
        heapq.heapify(unvisited_queue)

def mark_locations(array):
    #To get all 13 locations marked
    locations = array[['Latitude', 'Longitude']]
    fig = gmaps.figure()
    markers = gmaps.marker_layer(locations)
    fig.add_layer(markers)
    print("Locations Marked")
    return fig

def print_shortest(from_loc,to_loc):
    from_latlon = nom.geocode(from_loc)
    to_latlon = nom.geocode(to_loc)
    from_place = (from_latlon.latitude, from_latlon.longitude)
    to_place = (to_latlon.latitude, to_latlon.longitude)
    #To get mid points
    results = []
    with open("shortest_path.csv") as csvfile:
        reader = csv.reader(csvfile) # change contents to floats
        for row in reader: # each row is a list
            results.append(row)
    mid_point = []
    i = 3
    for s in results:
        if i > 2 and i < len(results)-2:
            result21 = nom.geocode(results[i])
            temp_place = (result21.latitude, result21.longitude)
            mid_point.append(temp_place)
            i+=1
        else:
            break
    fig = gmaps.figure()
    from2dest = gmaps.directions_layer(from_place, to_place,waypoints=mid_point)
    fig.add_layer(from2dest)
    print('Shortest path route')
    return fig

def get_Path(route,route2,number):
    d = len(route) -1
    i = 0
    transport_type=[]
    values = []
    for v in range(d):
        values.append(route[d-1].get_weight(route[d]) )
        d = d-1
    x=1
    for www in range(len(route) - 1):
        transport_type.append(route[www].transport[route[x]])
        x=x+1
    u = 0
    a = len(route2)-1
    path_toStr = ""
    path_toStr = path_toStr + route2[a] +" "

    for i in reversed(values):

```

```

        a = a-1
        path_toStr = path_toStr + " to "+route2[a]+" through "+transport_type[a]+" take
s "+ str(i) +" minutes, "
        u = i

    path_toStr = path_toStr + "Total timing for shortest path is "+ str(number)+ " minu
tes"

    return path_toStr

import heapq

if __name__ == '__main__':

    g = Graph()

    df1["Location"].apply(g.add_vertex)

    g.add_edge("University of Malaya", "PPUM", "Walking", 10)
    g.add_edge("University of Malaya" , "Bandar Utama Bus Hub", "Taxi", 15)

    g.add_edge("PPUM" , "Phileo Damansara","Bus T815", 4)
    g.add_edge("PPUM" , "Mid valley", "Bus734/T790", 7)

    g.add_edge("Phileo Damansara" , "Mutiara Damansara", "MRT Kajang/Sungai Buluh", 7)

    g.add_edge("Mutiara Damansara" , "Bandar Utama Bus Hub","Bus T809", 18)

    g.add_edge("Bandar Utama Bus Hub" , "Pasar Besar Larkin","Bus Billion Stars", 270)
    g.add_edge("Bandar Utama Bus Hub" , "Johor Larkin","Bus Five Stars", 250)

    g.add_edge("Pasar Besar Larkin" , "Johor Larkin","Walking", 1)

    g.add_edge("Johor Larkin" , "JB Customs","Bus Causeway link CW2", 14)
    g.add_edge("Johor Larkin" , "Larkin Bus Terminal","Walking", 20)
    g.add_edge("Johor Larkin" , "JB Sentral","Bus 227", 21)

    g.add_edge("Larkin Bus Terminal" , "JB Railway Sentral","Bus BET6", 11)

    g.add_edge("JB Customs" , "JB Sentral", "Walking", 10)

    g.add_edge("JB Railway Sentral" , "JB Sentral","Walking", 18)

    g.add_edge("Mid valley" , "Bandar Tasik Selatan","KTM" , 11)

    g.add_edge("Bandar Tasik Selatan" , "Terminal Bersepadu Selatan","Walking", 15)

    g.add_edge("Terminal Bersepadu Selatan" , "Johor Larkin","Bus billion Stars", 180)

    from_loc = 'University of Malaya'

    to_loc = 'JB Sentral'

    dijkstra(g, g.get_vertex(from_loc))
    target = g.get_vertex(to_loc)

    path = [target.get_id()]

```

```

coster = [target.get_distance()]

transport_type=[]

shortest(target, path,coster,transport_type)
x=1
for www in range(len(path) - 1):
    transport_type.append(g.get_vertex(path[x]).transport[g.get_vertex(path[www])])
    x=x+1

with open('shortest_path.csv','w') as f:
    thewriter = csv.writer(f,delimiter='\n')
    thewriter.writerow(["Location"])
    thewriter.writerow(path[::-1])

path_sp = ""
u = 0
a = len(path)-1
print(path[a], end='')

for i in reversed(coster[:-1]):
    #1 i = 10
    #2 i = 17
    #3 i = 28
    u = i - u
    #1 u = 10 - 0
    #2 u = 17 - 10
    #3 u = 28 - 17
    a = a-1
    print("to",path[a],"through",transport_type[a]," takes ", u , " minutes, " ,end
= '')
    u = i
    #1 u = 10
    #2 u = 17
    #3 u = 28

print()
print("Total timing for shortest path is ", target.get_distance(), " minutes")

path_sp = path_sp + "Total timing for shortest path is "+ str(target.get_distance
())+ " minutes"

route_1_path = [g.get_vertex("University of Malaya"),g.get_vertex("PPUM"),g.get_ver
tex("Phileo Damansara"),g.get_vertex("Mutiara Damansara"),
                g.get_vertex("Bandar Utama Bus Hub"),g.get_vertex("Pasar Besar Larkin")
                ,g.get_vertex("Johor Larkin"),g.get_vertex("JB Customs"),g.get_vertex("JB
Sentral")]
route_1_ready = ["University of Malaya","PPUM","Phileo Damansara","Mutiara Damansar
a","Bandar Utama Bus Hub","Pasar Besar Larkin","Johor Larkin","JB Customs","JB Sentral"
]

route_2_path = [g.get_vertex("University of Malaya"),g.get_vertex("PPUM"),g.get_ver

```

```

tex("Phileo Damansara"),g.get_vertex("Mutiar a Damansara"),
    g.get_vertex("Bandar Utama Bus Hub"),g.get_vertex("Pasar Besar Larkin")
    ,g.get_vertex("Johor Larkin"),g.get_vertex("Larkin Bus Terminal"),g.get_v
ertex("JB Railway Sentral"),g.get_vertex("JB Sentral")]

    route_2_ready = ["University of Malaya","PPUM","Phileo Damansara","Mutiar a Damansar
a","Bandar Utama Bus Hub","Pasar Besar Larkin","Johor Larkin","Larkin Bus Terminal","JB
Railway Sentral","JB Sentral"]

    route_3_path = [g.get_vertex("University of Malaya"),g.get_vertex("PPUM"),g.get_ver
tex("Mid valley"),g.get_vertex("Bandar Tasik Selatan"),
    g.get_vertex("Terminal Bersepadu Selatan"),g.get_vertex("Johor Larkin")
    ,g.get_vertex("JB Sentral")]

    route_3_ready = ["University of Malaya","PPUM","Mid valley","Bandar Tasik Selatan",
"Terminal Bersepadu Selatan","Johor Larkin","JB Sentral"]

    route_4_path = [g.get_vertex("University of Malaya"),g.get_vertex("PPUM"),g.get_ver
tex("Phileo Damansara"),g.get_vertex("Mutiar a Damansara"),
    g.get_vertex("Bandar Utama Bus Hub"),g.get_vertex("Johor Larkin")
    ,g.get_vertex("JB Customs"),g.get_vertex("JB Sentral")]

    route_4_ready = ["University of Malaya","PPUM","Phileo Damansara","Mutiar a Damansar
a","Bandar Utama Bus Hub","Johor Larkin","JB Customs","JB Sentral"]

    route_5_path = [g.get_vertex("University of Malaya"),g.get_vertex("Bandar Utama Bus
Hub"),g.get_vertex("Johor Larkin"),g.get_vertex("Larkin Bus Terminal"),
    g.get_vertex("JB Railway Sentral"),g.get_vertex("JB Sentral")]

    route_5_ready = ["University of Malaya","Bandar Utama Bus Hub","Johor Larkin","Lark
in Bus Terminal","JB Railway Sentral", "JB Sentral"]

    routes_path=[route_1_path,route_2_path,route_3_path,route_4_path,route_5_path]
    routes_ready=[route_1_ready,route_2_ready,route_3_ready,route_4_ready,route_5_ready
]

    routes_toStr=[]
    number = 0
    i = 0
    routes_score = []

    for r in range(len(routes_ready)):
        x=1
        number = 0
        for v in range(len(routes_ready[i])-1):
            number = number + routes_path[i][v].get_weight(routes_path[i][v+1])
            x = x+1

        routes_score.append(number)
        i = i+1

    for i in range(len(routes_path)):
        routes_toStr.append(get_Path(routes_path[i],routes_ready[i][::-1],routes_score[
i]))

    route_sp = target.get_distance()
    routes_path.append(route_sp)

    graph_df = pandas.read_csv("shortest_path.csv")

```

```
graph_df["Coordinates"] = graph_df["Location"].apply(nom.geocode, timeout = 10)
graph_df["Latitude"] = graph_df["Coordinates"].apply(lambda x: x.latitude if x != None else None)
graph_df["Longitude"] = graph_df["Coordinates"].apply(lambda x: x.longitude if x != None else None)
```

Dijkstra's shortest path

University of Malayato PPUM through Walking takes 10 minutes, to Mid valley through Bus734/T790 takes 7 minutes, to Bandar Tasik Selatan through KTM takes 11 minutes, to Terminal Bersepadu Selatan through Walking takes 15 minutes, to Johor Larkin through Bus billion Stars takes 180 minutes, to JB Sentral through Bus 227 takes 21 minutes,  
Total timing for shortest path is 244 minutes

In [5]:

```
#Mark All Locations Locations
#mark_Locations(df1)
```

In [6]:

```
#Mark Locations of shortest Route
#mark_Locations(graph_df)
```

In [7]:

```
#Shortes distance route
print_shortest(from_loc,to_loc)
```

Shortest path route



In [8]:

```
#Read the Stopwords file as an Array
stopwords = open("stopwords.txt", 'r')
stopwords = stopwords.read().splitlines()

positive = open("positive.txt", 'r')
positive = positive.read().splitlines()

negative = open("negative.txt", 'r')
negative = negative.read().splitlines()

#The rabin karp algorithm is made to filter stopwords from articles
def search(stopword, articles):
    s = len(stopword)
    a = len(articles)

    count = 0

    for i in range(0, a - s + 1):
        found = True
        for j in range(0, s):
            if stopwords[j] != articles[i + j]:
                found = False
                break
        if found:
            count += 1

    if count > 0:
        print('The stopwords', stopwords, 'has appeared for:', count, 'times!')
    else:
        None

def filter_stop(filepath):
    allArticles = open(filepath, encoding="utf8")
    allArticles = allArticles.read().splitlines()
    for x in allArticles:
        for i in stopwords:
            search(i, x)

def plotwordfreq(filepath):
    abc = open(filepath, encoding="utf8")
    abc = abc.read()
    number_of_characters = len(abc)
    print('Total number of words for the article:', number_of_characters, 'words')
    querywords = abc.split()
    resultwords = [word for word in querywords if word.lower() not in stopwords]

    d = {}
    wordsfreq = []
    for wordsfreq in resultwords:
        d[wordsfreq] = d.get(wordsfreq, 0)+1

    print()
    objectsaa = []
    objectsbb = []
    for key,value in d.items():
        objectsaa.append(key)
        objectsbb.append(value)
```

```

y_pos = np.arange(len(objectsaa))
plt.bar(y_pos, objectsbb)
plt.yticks(fontsize=30)
plt.xticks(y_pos, objectsaa)
plt.xticks(rotation = 90)
plt.xlabel('Value',fontsize=30)
plt.ylabel('Frequency',fontsize=30)
plt.rcParams['figure.figsize'] = (40,20)

```

```

def neg_pos(filepath):
    abc = open(filepath, encoding="utf8")
    abc = abc.read()
    negativeCount = 0
    positiveCount = 0
    number_of_characters = len(abc)

    for part in abc.split():
        if part in positive:
            positiveCount += 1
        elif part in negative:
            negativeCount += 1

    neutral = 0
    neutral = number_of_characters - (positiveCount+negativeCount)

    if positiveCount > negativeCount:
        print("The article is giving more positive sentiment")
    elif positiveCount < negativeCount:
        print("The article is giving more negative sentiment")
    else:
        print("The article is in neutral sentiment")

    print("Total number of Positivity is ",positiveCount)
    print("Total number of Negativity is ",negativeCount)
    print("Total number of Neutral words is ",neutral)

objects = ("Positive Words", "Negative Words")
y_pos = np.arange(len(objects))
values = [positiveCount, negativeCount]
plt.bar(y_pos, values, align='center', alpha=0.5, color = "blue")
plt.xticks(y_pos, objects)
plt.yticks(fontsize=10)
plt.xticks(fontsize=10)
plt.ylabel("Count",fontsize = 20)
plt.title("Positive & Negative words",fontsize = 20)
plt.rcParams['figure.figsize'] = (5,3)

```

In [9]:

```
filter_stop("all_types.txt")
```

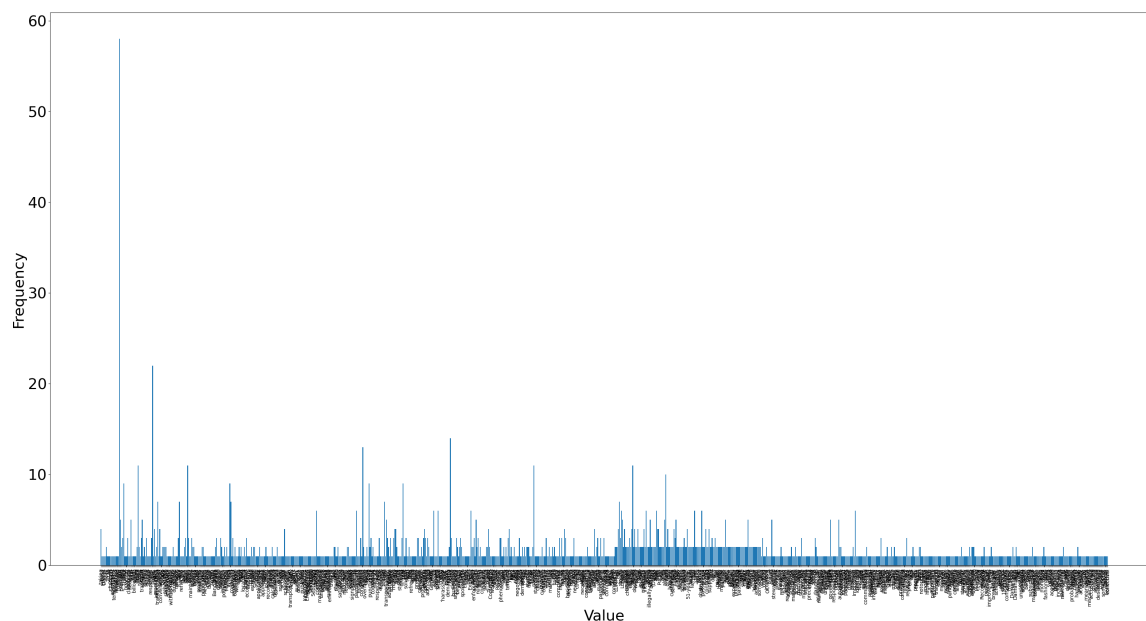
The stopword a has appeared for: 2386 times!  
The stopword about has appeared for: 9 times!  
The stopword above has appeared for: 1 times!  
The stopword after has appeared for: 11 times!  
The stopword again has appeared for: 10 times!  
The stopword against has appeared for: 8 times!  
The stopword all has appeared for: 60 times!  
The stopword am has appeared for: 67 times!  
The stopword an has appeared for: 364 times!  
The stopword and has appeared for: 149 times!  
The stopword any has appeared for: 9 times!  
The stopword are has appeared for: 69 times!  
The stopword aren't has appeared for: 1 times!  
The stopword as has appeared for: 229 times!  
The stopword at has appeared for: 296 times!  
The stopword be has appeared for: 99 times!  
The stopword because has appeared for: 6 times!  
The stopword been has appeared for: 13 times!  
The stopword before has appeared for: 4 times!  
The stopword being has appeared for: 4 times!  
The stopword between has appeared for: 16 times!  
The stopword both has appeared for: 2 times!  
The stopword but has appeared for: 19 times!  
The stopword by has appeared for: 19 times!  
The stopword could has appeared for: 8 times!  
The stopword did has appeared for: 4 times!  
The stopword do has appeared for: 25 times!  
The stopword does has appeared for: 1 times!  
The stopword doing has appeared for: 2 times!  
The stopword down has appeared for: 5 times!  
The stopword during has appeared for: 12 times!  
The stopword each has appeared for: 11 times!  
The stopword few has appeared for: 7 times!  
The stopword for has appeared for: 103 times!  
The stopword from has appeared for: 29 times!  
The stopword further has appeared for: 5 times!  
The stopword had has appeared for: 17 times!  
The stopword has has appeared for: 23 times!  
The stopword have has appeared for: 25 times!  
The stopword having has appeared for: 1 times!  
The stopword he has appeared for: 640 times!  
The stopword her has appeared for: 72 times!  
The stopword here has appeared for: 41 times!  
The stopword here's has appeared for: 1 times!  
The stopword hers has appeared for: 2 times!  
The stopword him has appeared for: 7 times!  
The stopword his has appeared for: 36 times!  
The stopword how has appeared for: 8 times!  
The stopword i has appeared for: 1897 times!  
The stopword if has appeared for: 31 times!  
The stopword in has appeared for: 562 times!  
The stopword into has appeared for: 3 times!  
The stopword is has appeared for: 154 times!  
The stopword it has appeared for: 224 times!  
The stopword its has appeared for: 19 times!  
The stopword me has appeared for: 168 times!  
The stopword more has appeared for: 18 times!  
The stopword most has appeared for: 13 times!  
The stopword my has appeared for: 7 times!  
The stopword no has appeared for: 97 times!  
The stopword nor has appeared for: 11 times!

The stopword not has appeared for: 49 times!  
The stopword of has appeared for: 156 times!  
The stopword off has appeared for: 20 times!  
The stopword on has appeared for: 350 times!  
The stopword once has appeared for: 3 times!  
The stopword only has appeared for: 20 times!  
The stopword or has appeared for: 264 times!  
The stopword other has appeared for: 13 times!  
The stopword ought has appeared for: 5 times!  
The stopword our has appeared for: 42 times!  
The stopword out has appeared for: 46 times!  
The stopword over has appeared for: 20 times!  
The stopword own has appeared for: 11 times!  
The stopword same has appeared for: 3 times!  
The stopword she has appeared for: 4 times!  
The stopword should has appeared for: 10 times!  
The stopword so has appeared for: 75 times!  
The stopword some has appeared for: 20 times!  
The stopword such has appeared for: 2 times!  
The stopword than has appeared for: 16 times!  
The stopword that has appeared for: 55 times!  
The stopword the has appeared for: 486 times!  
The stopword their has appeared for: 26 times!  
The stopword them has appeared for: 14 times!  
The stopword themselves has appeared for: 1 times!  
The stopword then has appeared for: 3 times!  
The stopword there has appeared for: 22 times!  
The stopword these has appeared for: 6 times!  
The stopword they has appeared for: 20 times!  
The stopword this has appeared for: 20 times!  
The stopword those has appeared for: 5 times!  
The stopword through has appeared for: 8 times!  
The stopword to has appeared for: 324 times!  
The stopword too has appeared for: 6 times!  
The stopword under has appeared for: 5 times!  
The stopword up has appeared for: 41 times!  
The stopword very has appeared for: 8 times!  
The stopword was has appeared for: 47 times!  
The stopword we has appeared for: 100 times!  
The stopword were has appeared for: 34 times!  
The stopword what has appeared for: 4 times!  
The stopword when has appeared for: 11 times!  
The stopword where has appeared for: 5 times!  
The stopword which has appeared for: 13 times!  
The stopword while has appeared for: 3 times!  
The stopword who has appeared for: 15 times!  
The stopword why has appeared for: 4 times!  
The stopword with has appeared for: 37 times!  
The stopword would has appeared for: 13 times!  
The stopword you has appeared for: 16 times!  
The stopword your has appeared for: 3 times!

In [28]:

```
plotwordfreq("Bus.txt")
```

Total number of words for the article: 17162 words



In [11]:

```
#plotwordfreq("Monorail.txt")
```

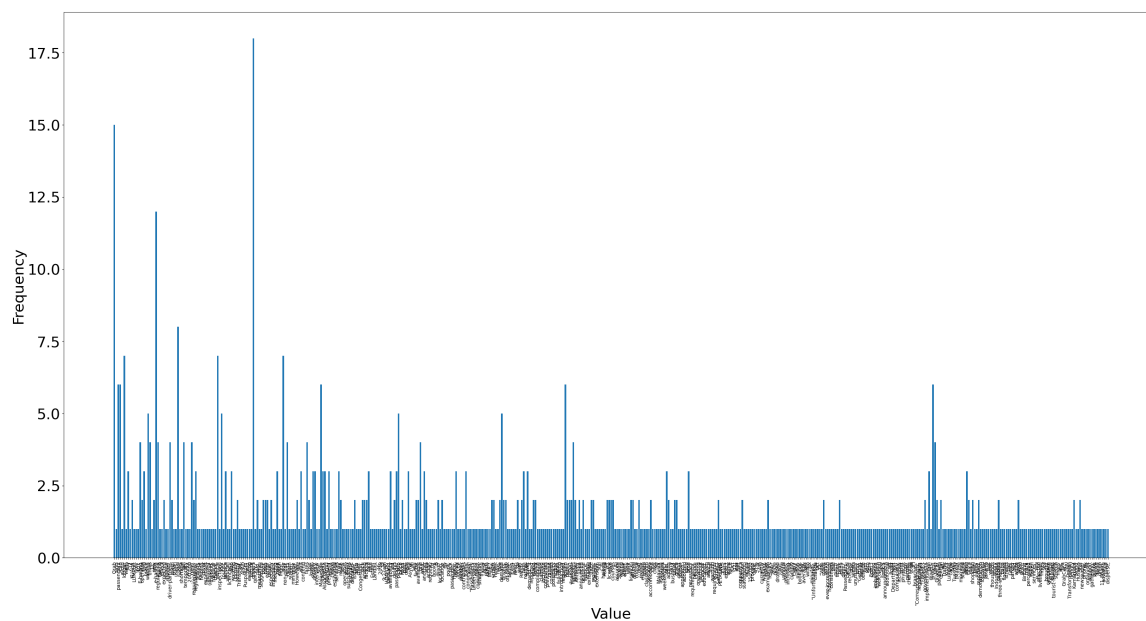
In [12]:

```
#plotwordfreq("KTM.txt")
```

In [13]:

```
plotwordfreq("Grab.txt")
```

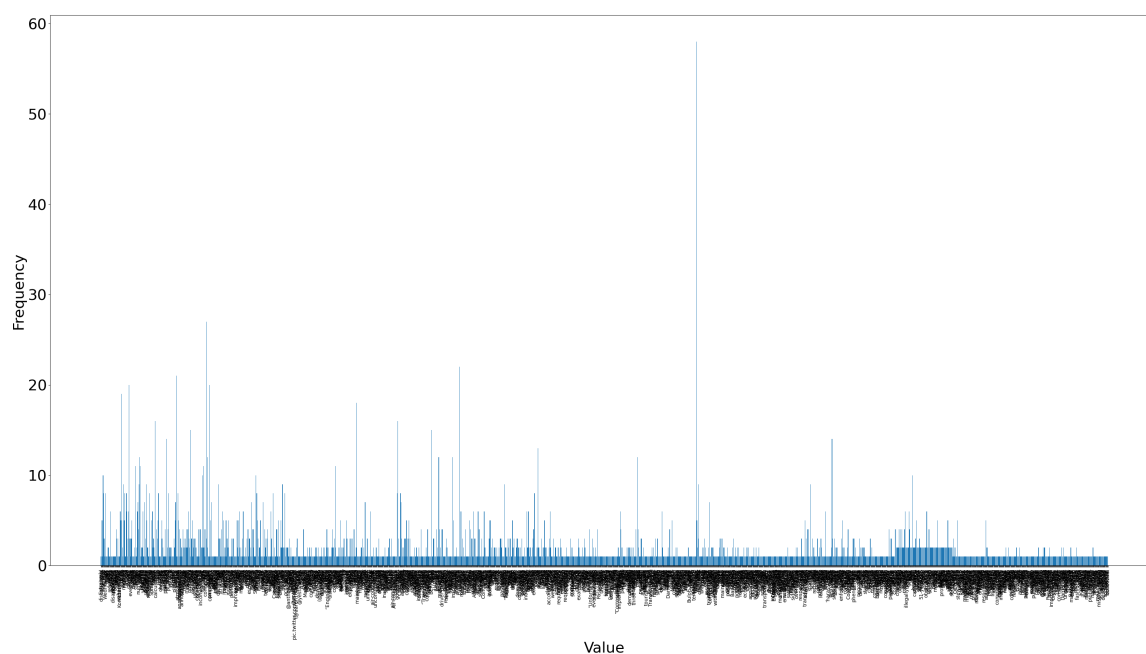
Total number of words for the article: 7544 words



In [14]:

```
plotwordfreq("all_types.txt")
```

Total number of words for the article: 34863 words



In [15]:

```
#plotwordfreq("LRT.txt")
```

In [26]:

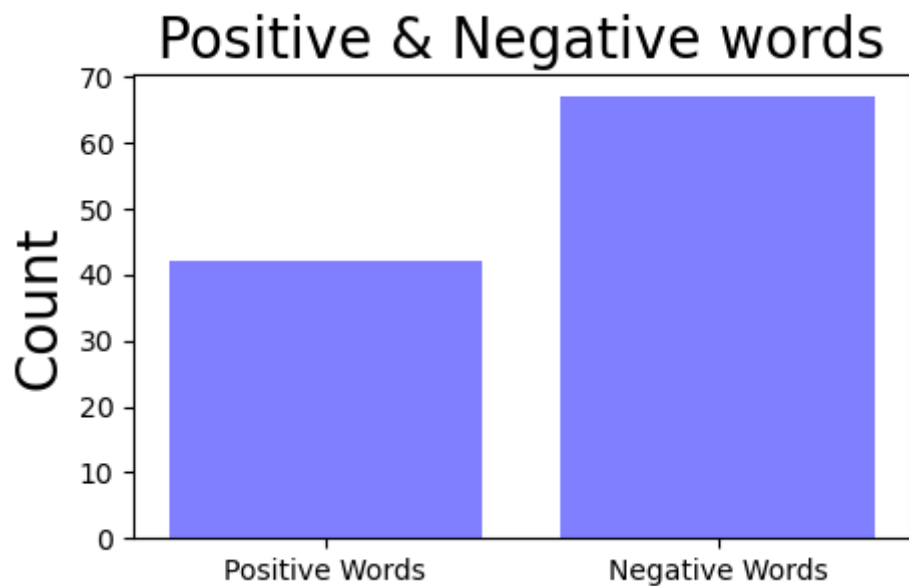
```
neg_pos("Bus.txt")
```

The article is giving more negative sentiment

Total number of Positivity is 42

Total number of Negativity is 67

Total number of Neutral words is 17053



In [17]:

```
#neg_pos("Monorail.txt")
```

In [18]:

```
#neg_pos("KTM.txt")
```

In [19]:

```
#neg_pos("LRT.txt")
```



In [20]:

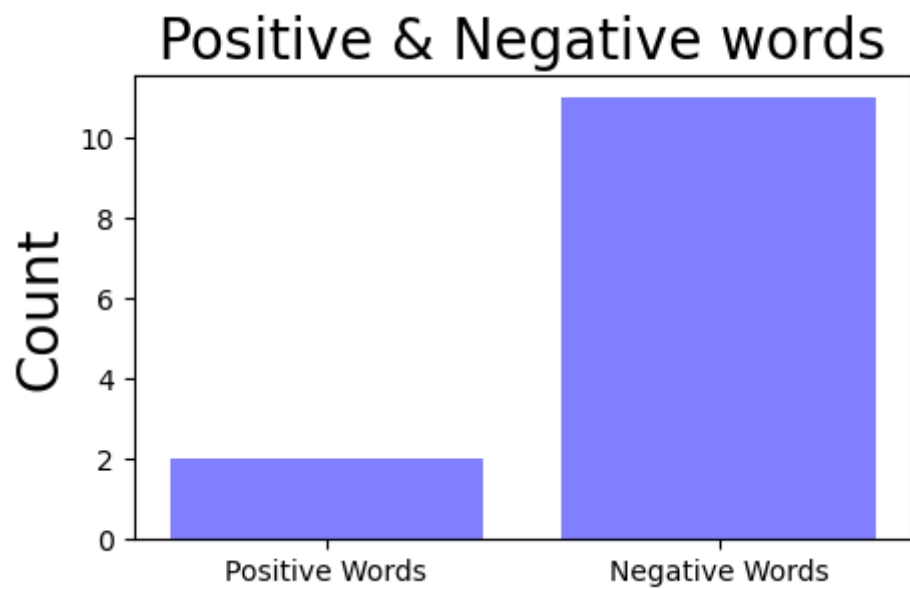
```
neg_pos("MRT.txt")
```

The article is giving more negative sentiment

Total number of Positivity is 2

Total number of Negativity is 11

Total number of Neutral words is 1222



In [21]:

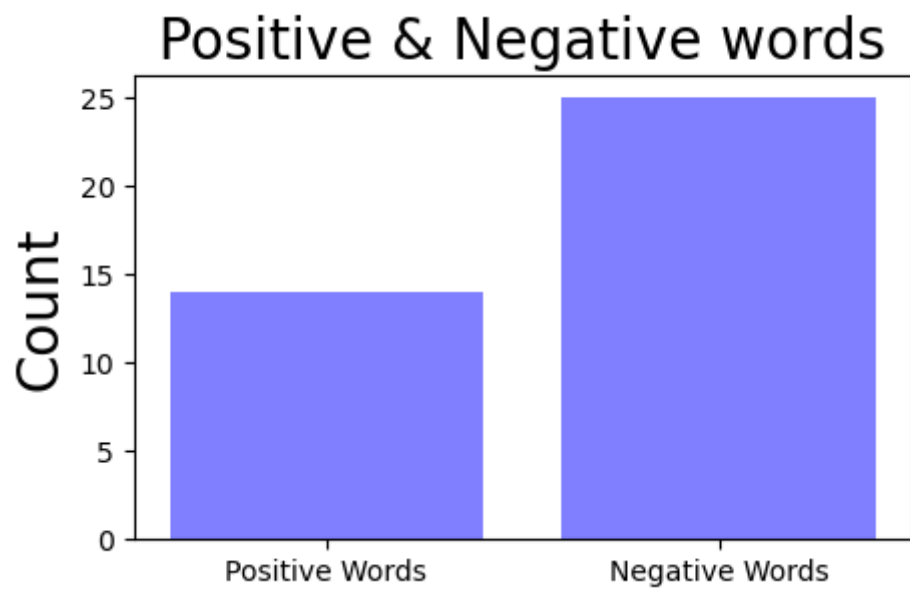
```
neg_pos("Grab.txt")
```

The article is giving more negative sentiment

Total number of Positivity is 14

Total number of Negativity is 25

Total number of Neutral words is 7505



In [22]:

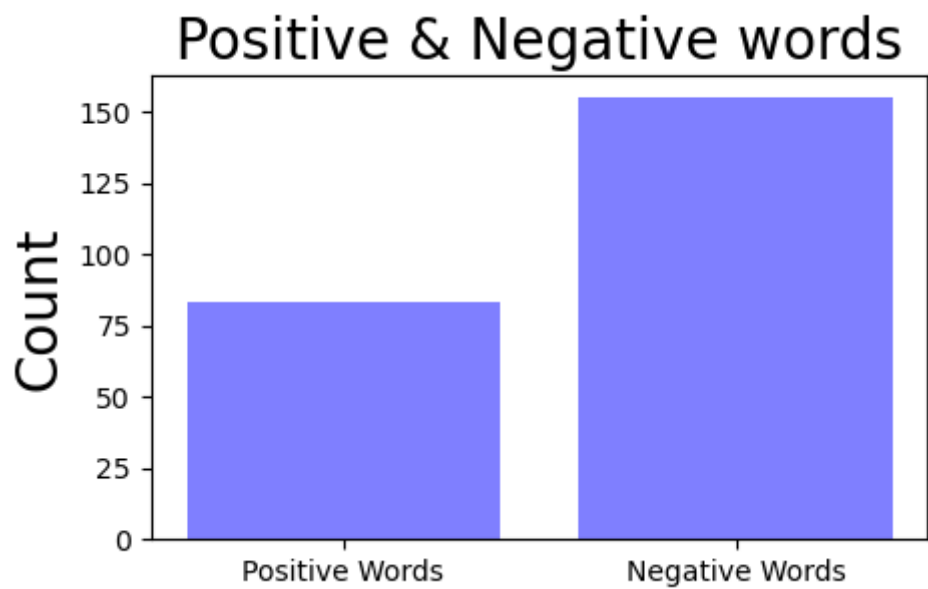
```
neg_pos("all_types.txt")
```

The article is giving more negative sentiment

Total number of Positivity is 83

Total number of Negativity is 155

Total number of Neutral words is 34625



In [23]:

```
def calc_sent(filepath):
    abc = open(filepath, encoding="utf8")
    abc = abc.read()
    negativeCount = 0
    positiveCount = 0
    number_of_characters = len(abc)
    calcCount = 0

    for part in abc.split():
        if part in positive:
            positiveCount += 1
        elif part in negative:
            negativeCount += 1

    calcCount = negativeCount - positiveCount

    return calcCount

route_1_score = calc_sent("MRT.txt") + calc_sent("Bus.txt") + routes_score[0]
route_2_score = calc_sent("MRT.txt") + calc_sent("Bus.txt") + routes_score[1]
route_3_score = calc_sent("KTM.txt") + calc_sent("Bus.txt") + routes_score[2]
route_4_score = calc_sent("MRT.txt") + calc_sent("Bus.txt") + routes_score[3]
route_5_score = calc_sent("Grab.txt") + calc_sent("Bus.txt") + routes_score[4]

scores = [route_1_score, route_2_score, route_3_score, route_4_score, route_5_score]
best = min(scores)
index = scores.index(best)

print("The best path according to the shortest path distance and the sentiment",
      "analysis based on articles found on the web is: \n \n", routes_toStr[index], "an",
      "d through the neagtivity",
      "and the postivity of the sentiment we have added their scores to the timing leng",
      "th so that the result will be" ,
      scores [index])
```

The best path according to the shortest path distance and the sentiment analysis based on articles found on the web is:

University of Malaya to PPUM through Bus 227 takes 10 minutes, to Mid valley through Bus billion Stars takes 7 minutes, to Bandar Tasik Selatan through Walking takes 11 minutes, to Terminal Bersepadu Selatan through KTM takes 15 minutes, to Johor Larkin through Bus734/T790 takes 180 minutes, to JB Sentral through Walking takes 21 minutes, Total timing for shortest path is 244 minutes and through the neagtivity and the postivity of the sentiment we have added their scores to the timing length so that the result will be 270

In [ ]: