

Question 1: A company sells n different products or items. The n -vector p gives the profit, in dollars per unit, for each of the n items. (The entries of p are typically positive, but a few items might have negative entries. These items are called loss leaders, and are used to increase customer engagement in the hope that the customer will make other, profitable purchases.) The n -vector s gives the total sales of each of the items, over some period (such as a month), i.e., s_i is the total number of units of item i sold. (These are also typically nonnegative, but negative entries can be used to reflect items that were purchased in a previous time period and returned in this one.) Express the total profit in terms of p and s using vector notation.

Solution: Created 2 lists profits and sales that takes input from the user. Converted the 2 lists to numpy array. To compute the total Profit, np.dot library is used which creates the dot product between the numpy arrays

```
import numpy as np

n = int(input("Enter the number of elements: "))
p = []
s = []

print("Enter the profits: ")
for i in range(n):
    a = int(input())
    p.append(a)

print("Enter the number of respective items: ")
for i in range(n):
    b = int(input())
    s.append(b)

profits = np.array(p)
sales = np.array(s)

res = np.dot(profits, sales)
print("The total Profit is " + str(res) + " Dollars")
```

```
Enter the number of elements: 4
Enter the profits:
2
5
-2
6
Enter the number of respective items:
5
2
0
7
The total Profit is 62 Dollars
```

Question 2: Just analyse use of vectors in word count problems..just write report or implementation detail.

Installing the textdistance python library

```
pip install textdistance
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting textdistance
  Downloading textdistance-4.5.0-py3-none-any.whl (31 kB)
Installing collected packages: textdistance
Successfully installed textdistance-4.5.0
```

Importing all necessary libraries

```
import pandas as pd
import numpy as np
import textdistance
import re
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
```

Extracting all the words from the given document using re library and storing it in the words list.

```

words = []
with open('data.txt', 'r') as f:
    file_data = f.read()
    file_data = file_data.lower()
    words = re.findall('\w+', file_data)

```

Creating a list V which stores the words without any duplicates. Set function is used to avoid the duplicates

```

#the list of all words
V = set(words)
words

```

```

['the',
 'project',
 'gutenberg',
 'ebook',
 'of',
 'moby',
 'dick',
 'or',
 'the',
 'whale',
 'by',
 'herman',
 'melville',
 'this',
 'ebook',
 'is',
 'for',
 'the',
 'use',
 'of',
 'anyone',
 'anywhere',
 'at',
 'no',
 'cost',
 'and',
 'with',
 'almost',
 'no',
 'restrictions',
 'whatsoever',
 'you',
 'may',
 'copy',
 'it',
 'give',
 'it',
 'away',
 'or',
 're',
 'use',
 'it',
 'under',
 'the',
 'terms',
 'of',
 'the',
 'project',
 'gutenberg',
 'license',
 'included',
 'with',
 'this',
 'ebook',
 'or',
 'online',
 'at',
 'www',

```

Printing the total number of unique words

```

print("Total number of unique wrds are: ", len(V))

Total number of unique wrds are: 420

```

from sklearn.feature_extraction.text, CountVectorizer is imported that gives the word a unique ID, with which the word is accessible.

```

from sklearn.feature_extraction.text import CountVectorizer
word_list = CountVectorizer(stop_words='english')
word_list.fit_transform(V)

<420x371 sparse matrix of type '<class 'numpy.int64'>'
  with 371 stored elements in Compressed Sparse Row format>

```

The words with it's unique ID are Printed

```

vocab = word_list.vocabulary_
vocab

```

```

{'34': 63,
 'ambergris': 135,
 '81': 110,
 'scenes': 309,
 'end': 200,
 'works': 369,
 'december': 188,
 '109': 10,
 'pitchpoling': 288,
 '68': 97,
 'chase': 168,
 '36': 65,
 'honor': 233,
 'dick': 191,
 'rachel': 300,
 'squid': 327,
 '97': 126,
 'etymology': 205,
 'cost': 177,
 'crew': 179,
 'david': 185,
 'postscript': 289,
 'enter': 202,
 '91': 120,
 '2008': 47,
 'candles': 158,
 'skeleton': 319,
 'quadrant': 296,
 'nightgown': 278,
 '126': 29,
 'shark': 315,
 '58': 87,
 '72': 101,
 '121': 24,
 'armada': 137,
 'fish': 209,
 'deck': 189,
 'org': 281,
 'whiteness': 366,
 'ahab': 132,
 'bachelor': 142,
 'heidelburgh': 229,
 '79': 108,
 '50': 79,
 '59': 88,
 '129': 32,
 'needle': 276,
 '132': 36,
 '134': 38,
 'whales': 362,
 '57': 86,
 '41': 70,
 'iron': 237,
 '25': 53,
 '48': 77,
 '47': 76,
 '52': 81,
 'gilder': 218,

```

```

vocab['moby']

```

```

270

```

Counting the frequency of each word and storing it as a dictionary and printing it

```
#word frequency
word_freq = {}
word_freq = Counter(words)
print(word_freq.most_common()[0:50])

[('chapter', 135), ('the', 106), ('and', 18), ('of', 15), ('whale', 15), ('in', 11), ('s', 9), ('a', 8), ('or', 7), ('ahab', 6), ('eboo
```

Calculating the relative frequency of each word

```
#relative frequency
probability = {}
total_freq = sum(word_freq.values())
for i in word_freq.keys():
    probability[i] = word_freq[i]/total_freq
```

Building the function for the auto-correction, that evaluates the given word, if found wrong then it formulates the nearest possible correct word


```
def my_autocorrect(value):
    value = value.lower()
    if value in V:
        return ('Your word is absolutely Perfect')
    else:
        sim = [1-(textdistance.Jaccard(qval=2).distance(v, value)) for v in word_freq.keys()]
        df = pd.DataFrame.from_dict(probability, orient='index').reset_index()
        df = df.rename(columns={'index':'Word', 0:'Prob'})
        df['Similarity'] = sim
        output = df.sort_values(['Similarity', 'Prob'], ascending=False).head()
        return(output)
```

```
text = input("Enter the word to check for correction: ")
my_autocorrect(text)
```

```
Enter the word to check for correction: project
'Your word is absolutely Perfect'
```

```
text = input("Enter the word to check for correction: ")
my_autocorrect(text)
```

```
Enter the word to check for correction: Chaptr
```

	Word	Prob	Similarity	
74	chapter	0.164034	0.571429	
89	chapel	0.001215	0.428571	
413	chase	0.003645	0.285714	
176	chart	0.001215	0.285714	
55	character	0.001215	0.181818	

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.