```
# Import the pandas library to facilitate data manipulation and analysis
import pandas as pd

# Import the CountVectorizer class from Scikit-learn for text analysis
from sklearn.feature_extraction.text import CountVectorizer

# Create an instance of CountVectorizer to convert text data into token count matrices
count_vect = CountVectorizer()

from IPython.core.display import display_html
# Define two distinct text documents for analysis
Document1 = """Data types are fundamental building blocks in computer programming and data representation.They define the kind of
Document2 = """Data types are an essential concept in computer programming and data management. They define the nature of data an

# Create a corpus by collecting the defined documents
corpus=[Document1,Document2]

# Convert the text data into a token count matrix
matrix1 = count_vect.fit_transform(corpus)
print(matrix1)
```

```
  (0, 21)       8
  (0, 102)      5
  (0, 4)        3
  (0, 36)       1
  (0, 9)        1
  (0, 7)        1
  (0, 42)       3
  (0, 18)       1
  (0, 73)       1
  (0, 3)        7
  (0, 79)       1
  (0, 97)       1
  (0, 22)       1
  (0, 95)       3
  (0, 51)       1
  (0, 62)       3
  (0, 94)       2
  (0, 10)       3
  (0, 6)        1
  (0, 88)       1
  (0, 107)      2
  (0, 67)       3
  (0, 58)       2
  (0, 54)       1
  (0, 24)       1
  :     :
  (1, 40)       1
  (1, 59)       1
  (1, 71)       1
  (1, 85)       1
  (1, 12)       1
  (1, 90)       1
  (1, 5)        1
  (1, 72)       1
  (1, 109)      1
  (1, 41)       1
  (1, 1)        1
  (1, 74)       1
  (1, 81)       1
  (1, 28)       1
  (1, 47)       1
  (1, 66)       1
  (1, 27)       1
  (1, 17)       1
  (1, 55)       1
  (1, 96)       1
  (1, 20)       1
  (1, 26)       1
  (1, 80)       1
  (1, 77)       1
  (1, 92)       1
```

```
# Convert the matrix to an array for easier manipulation
Mat01=matrix1.toarray()

# Display the shape of the new array to understand its dimensions
Mat01.shape

# Create a vector that represents token count for the first and second documents
vecA=Mat01[0,:]
vecB=Mat01[1,:]
print(vecA)
print(vecB)
```

```
[1 0 0 7 3 0 1 1 2 1 3 0 0 4 1 1 0 0 1 0 0 8 1 1 1 1 0 0 0 1 0 1 2 1 1 1 1
 0 1 2 0 0 3 1 1 1 1 0 0 1 0 1 0 0 1 0 0 1 2 0 0 3 3 0 1 0 0 3 1 0 2 0 0 1
 0 1 1 0 2 1 0 0 1 1 1 0 1 1 1 2 0 1 0 1 2 3 0 1 1 1 1 1 5 1 1 1 2 2 0 0 1
 1 1]
[ 0  1  1 10  1  1  1  0  1  1  1  1  1  1  0  1  1  1  1  1  1 10  1  0
  1  1  1  1  1  1  1  0  1  0  0  0  0  1  1  0  1  1  4  1  0  0  1  1
  1  0  1  0  1  1  0  1  1  1  3  1  1  1  2  2  0  1  1  1  0  1  1  1
  1  2  1  0  1  1  0  0  1  1  0  1  1  1  0  1  0  1  1  0  1  0  1  5
  1  1  1  0  0  1  4  1  1  0  0  0  1  1  0  0  0]
```

```
# Convert the text data into a token count matrix
matrix1 = count_vect.fit_transform(corpus)
print(matrix1)
```

```
  (0, 21)       8
  (0, 102)      5
  (0, 4)        3
  (0, 36)       1
  (0, 9)        1
  (0, 7)        1
  (0, 42)       3
  (0, 18)       1
  (0, 73)       1
  (0, 3)        7
  (0, 79)       1
  (0, 97)       1
  (0, 22)       1
  (0, 95)       3
  (0, 51)       1
  (0, 62)       3
  (0, 94)       2
  (0, 10)       3
  (0, 6)        1
  (0, 88)       1
  (0, 107)      2
  (0, 67)       3
  (0, 58)       2
  (0, 54)       1
  (0, 24)       1
  :     :
  (1, 40)       1
  (1, 59)       1
  (1, 71)       1
  (1, 85)       1
  (1, 12)       1
  (1, 90)       1
  (1, 5)        1
  (1, 72)       1
  (1, 109)      1
  (1, 41)       1
  (1, 1)        1
  (1, 74)       1
  (1, 81)       1
  (1, 28)       1
  (1, 47)       1
  (1, 66)       1
  (1, 27)       1
  (1, 17)       1
  (1, 55)       1
  (1, 96)       1
  (1, 20)       1
  (1, 26)       1
  (1, 80)       1
  (1, 77)       1
  (1, 92)       1
```

```
import numpy as np
from numpy import linalg as LA

# Calculate the dot product of vectors
vdot=np.dot(vecA,vecB)

# Calculate the Euclidean norm of each vector
vnorm1=LA.norm(vecA,2)
vnorm2=LA.norm(vecB,2)

# Compute the cosine similarity
vcos=vdot/(vnorm1*vnorm2)
print('cosine angle:', vcos)

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Create a TfidfVectorizer object
vectorizer=TfidfVectorizer()

# Apply the a method to convert the 'corpus' using TF-IDF vectorization.
trsfm=vectorizer.fit_transform(corpus)

# Generate a visualization of the new token count matrix similar to step one
pd.DataFrame(trsfm.toarray(),columns=count_vect.get_feature_names_out(),index=['Document 1', 'Document 2'])
```

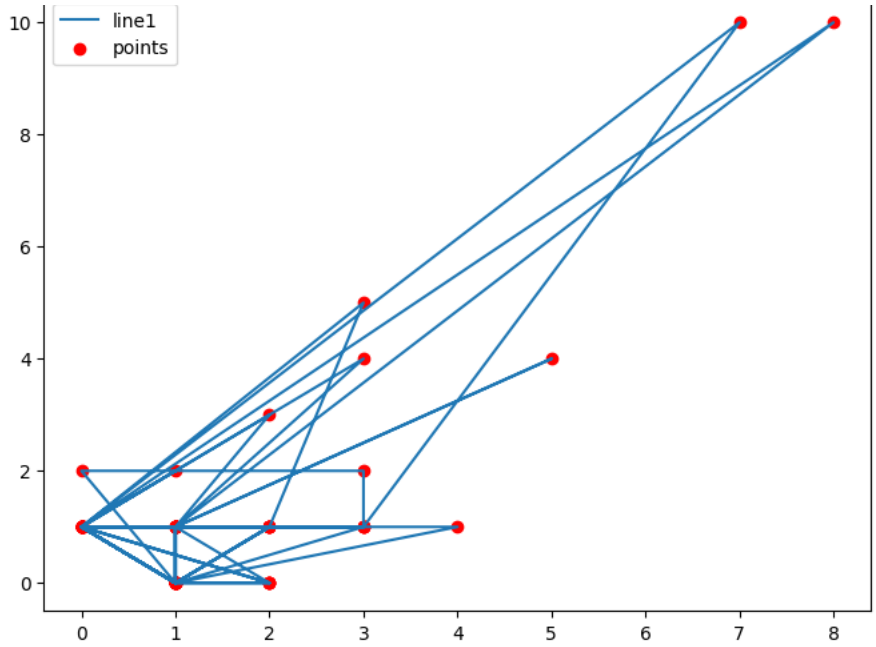
```
cosine angle: 0.7869219042760625
```

	accurate	allocation	an	and	are	as	be	bloc
Document 1	0.074923	0.000000	0.000000	0.373160	0.159926	0.000000	0.053309	0.0749
Document 2	0.000000	0.070994	0.070994	0.505131	0.050513	0.070994	0.050513	0.0000

2 rows × 113 columns

```
# Transform token count matrix into array and then extract vector for each document
Mat02 = trsfm.toarray()
vec01=Mat02[0,:]
vec02=Mat02[1,:]
# Calculate the dot product similarity between two vector
vdot=np.dot(vec01, vec02)

cosine_sim = cosine_similarity(trsfm[0,:],trsfm)
print('cosine similarity is:', cosine_sim)

import numpy as np
from numpy import linalg as la

# Normalize the vectors
norm1 = np.linalg.norm(vecA,2)
norm2 = np.linalg.norm(vecB,2)
NormVec1 = vecA/norm1
NormVec2 = vecB/norm2

# Calculate the absolute difference between the normalized vectors
ABS_DIFF = np.abs(vecA - vecB)
print(ABS_DIFF)

# Calculate the norm of this new difference vector
AH = np.linalg.norm(NormVec1 - NormVec2)
AG = np.linalg.norm(ABS_DIFF)

# Print the absolute difference and euclidean difference between the two vectors
print(AH,AG)


import matplotlib.pyplot as plt

# Calculate the absolute difference in word count between the two vectors
absolute_difference = np.abs(vecA - vecB)
print("absolute difference vector",absolute_difference)
print(np.sum(absolute_difference))

# Get the feature names (words) in alphabetical order
vectorizer.get_feature_names_out(Document1)
```

```
vectorizer.get_feature_names_out(Document1)
vectorizer.get_feature_names_out(Document2)

# Create a figure with two subplots
plt.figure(figsize=(8,6))
plt.plot(vecA,vecB, label='line1')
plt.scatter(vecA, vecB, label='points', color='red')
plt.legend()
plt.show()

# Create the first subplot for first vector and plot
plt.subplot(2, 1, 1)
plt.plot(vecA)
plt.title('First Vector')
plt.xlabel('Index')
plt.ylabel('Value')

# Create a bar plot for absolute word count differences in vec01
plt.subplot(2, 2, 2)
plt.bar(range(len(absolute_difference)), absolute_difference)
plt.title('Absolute Difference in Word Count')
plt.ylabel('Absolute Difference')
plt.xlabel('Index')

# Create the second subplot second vector
plt.subplot(2, 2, 3)
plt.plot(vecB)
plt.title('Second Vector')
plt.xlabel('Index')
plt.ylabel('Value')

# Create a bar plot for absolute word count differences in vec02
plt.subplot(2, 2, 4)
plt.bar(range(len(absolute_difference)), absolute_difference)

plt.title('Absolute Difference in Word Count')
plt.xlabel('Index')
plt.ylabel('Absolute Difference')

plt.tight_layout()
```




```
<ipython-input-6-4afe90ecabf2>:58: MatplotlibDeprecationWarning: Auto-removal of
  plt.subplot(2, 2, 2)
```