Assignment-2 n-gram TF-IDF and document similarity

Objective

- Learn Term Frequency
- Compute document similarity

Your Details

```
import datetime

student_rollno = 24
student_name = 'Rachit Basnet'
assignment_tag = 'MDS555-2023-Assignment-2'
```

```
# from checker_utils import done
def done(task):
    _date = datetime.datetime.now()
    task = task + ": " + str(_date)
    print('='*len(task), '\n', task , '\n', '='*len(task), sep='')
    pass
```

Literature Review

- Put your review of the literature related to n-gram TF-IDF and document similarity
- define terminologies used
- put details of the library used

Task 1: Dataset Preparation:

Prepare the Nepali news dataset *(hint: you can obtain text from news websites, at least 20 different news of 2/3 different categories)*. Host the dataset in the public git repository.

```
# Task 1:Dataset preparation:
import pandas as pd
import pandas as pd
#!git clone https://github.com/rachitbasnet/Assignment.git
#Load dataset
df = pd.read_csv('https://raw.githubusercontent.com/rachitbasnet/Assignment/main/news%20for%20nlp3.csv')
print(df)
```

```
    Sn Catogory                                                News
0    1  Finance  मूल्यवृद्धिसँगै अब काठमाडौंमा पेट्रोल प्रतिलिट...
1    2  Finance  सरकारले पेट्रोलियम पदार्थको मूल्य फेरि बढाएको ...
2    3  Finance  हालसम्म ४ वाणिज्य बैंकहरुले गत आर्थिक वर्षको न...
3    4  Finance  चालु आर्थिक वर्षको पहिलो महिना साउनमा मुलुकबाट...
4    5  Opinion  नेपाली राजनीति र साहित्यमा सबैभन्दा धेरै एकैसा...
5    6  Opinion  पत्रकार रमेशकुमारले हिमालखबरमा अघिल्लो साता आक...
6    7  Opinion  बिरामीको उपचारमा लापरबाही गरेको आरोप लगाउँदै ब...
7    8  Opinion  नेपालको राजनीतिमा अचेल सबै ठूला दलका नेताहरू ए...
8    9  Finance  जिल्लाको उत्तरी भेगमा भएर बग्ने कालीगण्डकी नदी...
9   10  Finance   नेपाल धितोपत्र बोर्ड (सेबोन) ले ब्रोकर कमिसन ...
10  11  Finance   एक अर्ब रुपैयाँभन्दा बढी चुक्ता पुँजी भएका कम...
11  12  Finance  भारतको सबैभन्दा ठूलो वायुसेवा कम्पनी इन्डिगोले...
12  13  Finance  सरकारले निजी क्षेत्रलाई लगानीको वातावरण तयार प...
13  14  Finance  काठमाडौं तराई/मधेश द्रुतमार्ग (फास्ट ट्र्याक) ...
14  15  Finance  नेपाललाई मेला, सभा/सम्मेलन तथा विवाह गन्तव्यका...
15  16  Finance  लुम्बिनीमा ९ लाख ७४ हजार ३ सय ८१ हेक्टर वन क्ष...
16  17   Sports  एसियाली खेलकुदमा ई–स्पोर्ट्सले पहिलोपल्ट प्रवे...
17  18   Sports  चीनले फेरि एकपल्ट आफ्नो भूमिमा हुने १९ औं एसिय...
18  19   Sports  इजरायलको महिला फुटबल लिगमा हापोएल रानानाबाट खे...
19  20   Sports  पुलिसकी शुभाङ्गी श्रेष्ठले ११ औं कोरियन एम्बास...
```

```
done('Task 1')
```

```
==================================
Task 1: 2023-09-25 14:40:17.004515
==================================
```

Task 2.1: Prepare one-gram, bi-gram, tri-gram vocabulary

```
# Task 2.1:Frequency Analysis
import nltk
```

```
import string
from nltk.tokenize import word_tokenize
nltk.download('punkt')
# Function to tokenize and clean the 'Input' column
def tokenize_and_clean_text(df, input_column_name, new_column_name):
    def clean_text(text):
        # Remove punctuation and numbers, and convert to lowercase
        text = text.replace('।', '')
        text = text.replace('‘', '')
        text = text.replace('’', '')
        text = text.replace('–', '')

        text = ''.join([char for char in text if char not in string.punctuation and not char.isdigit()])
        return text.lower()

    df[new_column_name] = df[input_column_name].apply(clean_text)
    df[new_column_name] = df[new_column_name].apply(lambda x:word_tokenize(x))
    return df

# Tokenize and clean the 'Input' column, storing the result in a new column called 'Tokenized_Input'
df = tokenize_and_clean_text(df, 'News', 'Tokenized_Input')

df.head()
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

	Sn	Catogory	News	Tokenized_Input
0	1	Finance	मूल्यवृद्धिसँगै अब काठमाडौंमा पेट्रोल प्रतिलिट...	[मूल्यवृद्धिसँगै, अब, काठमाडौंमा, पेट्रोल, प्र...
1	2	Finance	सरकारले पेट्रोलियम पदार्थको मूल्य फेरि बढाएको ...	[सरकारले, पेट्रोलियम, पदार्थको, मूल्य, फेरि, ब...
2	3	Finance	हालसम्म ४ वाणिज्य बैंकहरुले गत आर्थिक वर्षको न...	[हालसम्म, वाणिज्य, बैंकहरुले, गत, आर्थिक, वर्ष...
			चालु आर्थिक वर्षको पहिलो महिना	[चालु, आर्थिक, वर्षको, पहिलो, महिना,

```
done('Task 1')
```

```
==================================
Task 1: 2023-09-25 05:48:56.442468
==================================
```

```
# Task 2: Prepare one-gram, bi-gram, tri-gram vocabulary
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
import nltk
from nltk import ngrams
from nltk.corpus import stopwords
#tokenize text
from nltk import FreqDist,word_tokenize

all_tokens = [token for tokens in df['Tokenized_Input'] for token in tokens]
all_tokens
```

```
 'आ',
 'कोरियन',
 'एम्बासडर',
 'खुला',
 'तेक्कान्दो',
 'प्रतियोगितामा',
 'स्वर्ण',
 'पदक',
 'जितेकी',
 'छन्',
 'जुनियर',
 'महिला',
 'केजीमुनि',
 'फाइनलमा',
 'उनले',
 'राजदेवी',
 'तेक्कान्दोकी',
 'प्रज्ञा',
 'परियारलाई',
 'पराजित',
 'गरिन्']
```

```
import nltk
nltk.download('stopwords')
#Stop word filter
from nltk.corpus import stopwords

#load list of nepali stop words
nepali_stop_words = set(stopwords.words('nepali'))
#print(nepali_stop_words)

filtered_words = [word for word in all_tokens if word not in nepali_stop_words]
filtered_words
```

```
 'पदक',
 'महत्त्वाकांक्षा',
 'अत्यधिक',
 'इजरायलको',
 'महिला',
 'फुटबल',
 'लिगमा',
 'हापोएल',
 'रानानाबाट',
 'खेलिरहेकी',
 'नेपाली',
 'महिला',
 'राष्ट्रिय',
 'टोलीकी',
 'स्ट्राइकर',
 'सावित्रा',
 'भण्डारी',
 'एसियाली',
 'खेलकुदका',
 'शनिबार',
 'काठमाडौं',
 'आइपुग्ने',
 'छिन्',
 'पुलिसकी',
 'शुभाङ्गी',
 'श्रेष्ठले',
 'कोरियन',
 'एम्बासडर',
 'खुला',
 'तेक्वान्दो',
 'प्रतियोगितामा',
 'स्वर्ण',
 'पदक',
 'जितेकी',
 'जुनियर',
 'महिला',
 'केजीमुनि',
 'फाइनलमा',
 'राजदेवी',
 'तेक्वान्दोकी',
 'प्रज्ञा',
 'परियारलाई',
 'पराजित',
 'गरिन्']
```

```
#create n-grams
one_gram = filtered_words
bi_gram =list(ngrams(filtered_words,2))
tri_gram = list(ngrams(filtered_words,3))
```

```
print("one grams",one_gram)
print("two grans",bi_gram)
print("tri-grams",tri_gram)
```

```
one grams ['मूल्यवृद्धिसँगै', 'काठमाडौंमा', 'पेट्रोल', 'प्रतिलिटर', 'डिजेलमट्टितेल', 'प्रतिलिटर', 'पुगेको', 'बढेको', 'मूल्य', 'शनिबार', 'बिहान', 'बजेदेखि', 'लागू', 'निगम
two grans [('मूल्यवृद्धिसँगै', 'काठमाडौंमा'), ('काठमाडौंमा', 'पेट्रोल'), ('पेट्रोल', 'प्रतिलिटर'), ('प्रतिलिटर', 'डिजेलमट्टितेल'), ('डिजेलमट्टितेल', 'प्रतिलिटर'), ('प्रतिलिटर'
tri-grams [('मूल्यवृद्धिसँगै', 'काठमाडौंमा', 'पेट्रोल'), ('काठमाडौंमा', 'पेट्रोल', 'प्रतिलिटर'), ('पेट्रोल', 'प्रतिलिटर', 'डिजेलमट्टितेल'), ('प्रतिलिटर', 'डिजेलमट्टितेल', 'प्रति
```

```
done('Task 2')
```

```
==================================
Task 2: 2023-09-25 14:40:42.101800
==================================
```

Task 3: Compute TF-IDF vectors for each vocabulary

```
from sklearn.feature_extraction.text import TfidfVectorizer


# Create a TfidfVectorizer for one-grams
tfidf_vectorizer_onegram = TfidfVectorizer()

# Fit and transform the documents for one-gram vocabulary
tfidf_matrix_onegram = tfidf_vectorizer_onegram.fit_transform(df['News'])

# Get the TF-IDF features and vocabulary for one-gram vocabulary
tfidf_features_onegram = tfidf_matrix_onegram.toarray()
tfidf_vocab_onegram = tfidf_vectorizer_onegram.get_feature_names_out()

# Now, tfidf_features_onegram contains the TF-IDF vectors for one-gram vocabulary
# tfidf_vocab_onegram contains the vocabulary

# Print or store TF-IDF features and vocabulary for each vocabulary as needed
print("One-gram TF-IDF Features:", tfidf_features_onegram)
print("One-gram TF-IDF Vocabulary:", tfidf_vocab_onegram)
```

```
One-gram TF-IDF Features: [[0.         0.         0.         ... 0.         0.12900128 0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.25425631 ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
```

```
 [0.         0.         0.         ... 0.         0.         0.        ]]
One-gram TF-IDF Vocabulary: ['अघ' 'अच' 'अझ' 'अत' 'अध' 'अन' 'अब' 'अभ' 'अर' 'अवस' 'अस' 'आइप' 'आईएमई'
 'आईप' 'आएक' 'आक' 'आकर' 'आज' 'आध' 'आफ' 'आम' 'आय' 'आयल' 'आर' 'आवश' 'इएक'
 'इकर' 'इजर' 'इड' 'इन' 'इनन' 'इनलम' 'इर' 'उक' 'उच' 'उठ' 'उत' 'उन' 'उनक'
 'उनम' 'उनल' 'उपच' 'उम' 'उर' 'उल' 'एउट' 'एक' 'एकअर' 'एकपल' 'एनआईस' 'एभर'
 'एम' 'एल' 'एव' 'एस' 'एसन' 'ऐन' 'ओप' 'कक' 'कथनम' 'कन' 'कब' 'कम' 'कर' 'करण'
 'करणक' 'करणम' 'करल' 'कल' 'कसर' 'कहर' 'खण' 'खर' 'गइरह' 'गक' 'गठन' 'गण'
 'गत' 'गन' 'गम' 'गमक' 'गमल' 'गर' 'घट' 'घटन' 'ङच' 'चतम' 'चन' 'चम' 'चर' 'छन'
 'जक' 'जगढसम' 'जद' 'जन' 'जलव' 'जस' 'ञप' 'टक' 'टन' 'टबल' 'टर' 'टरल' 'टरहर'
 'टव' 'ठक' 'ठकल' 'ठम' 'ठल' 'डक' 'डल' 'ढक' 'णय' 'णयम' 'तथ' 'तन' 'तप' 'तम'
 'तय' 'तर' 'तव' 'थक' 'थनम' 'थप' 'थम' 'दक' 'दन' 'दब' 'दम' 'दर' 'दलक' 'धन'
 'धब' 'धम' 'नक' 'नगर' 'नद' 'नमन' 'नय' 'नल' 'पकमल' 'पछ' 'पत' 'पद' 'पदक'
 'पन' 'पनक' 'पम' 'पर' 'परब' 'पल' 'पह' 'फत' 'बग' 'बज' 'बढ' 'बत' 'बन' 'बर'
 'बरक' 'बल' 'बस' 'भइसक' 'भई' 'भएक' 'भएर' 'भण' 'भद' 'भन' 'मक' 'मट' 'मध'
 'मन' 'मह' 'महत' 'यअन' 'यक' 'यटर' 'यध' 'यन' 'यम' 'यमम' 'यर' 'यरधन' 'यलक'
 'यव' 'यवस' 'यस' 'यसअघ' 'यसक' 'यसब' 'यसम' 'यसल' 'रक' 'रज' 'रण' 'रत' 'रतक'
 'रद' 'रध' 'रब' 'रम' 'रमण' 'रमणक' 'रमम' 'रमश' 'रय' 'रल' 'रव' 'रवक' 'रवर'
 'रष' 'रस' 'रसम' 'रह' 'लक' 'लकहर' 'लखबरम' 'लग' 'लन' 'लम' 'लल' 'लसम' 'वन'
 'वनक' 'वर' 'वरक' 'वरण' 'वरप' 'वल' 'शक' 'शन' 'शनब' 'षक' 'षकसह' 'षड' 'षण'
 'षद' 'षपछ' 'षम' 'षयल' 'षयवस' 'सक' 'सञ' 'सडर' 'सत' 'सध' 'सन' 'सनब' 'सब'
 'सभ' 'सम' 'समर' 'समस' 'सय' 'सरक' 'सल' 'सहकर' 'सहभ' 'हक' 'हज' 'हद' 'हम'
 'हर' '११' '१२' '१४' '१६' '१७३' '१८३' '१९' '१९७१' '२०३९' '२३' '२४' '२५'
 '३०' '३५' '४२' '४४' '५०' '५८' '५९' '६५' '७०' '७२' '७३' '७४' '८१' '८९'
 '९१' '९७७']
```

Double-click (or enter) to edit

```
#for Bigram

# Create a TfidfVectorizer for one-grams
tfidf_vectorizer_bigram = TfidfVectorizer()

# Fit and transform the documents for one-gram vocabulary
tfidf_matrix_bigram = tfidf_vectorizer_bigram.fit_transform(df['News'])

# Get the TF-IDF features and vocabulary for one-gram vocabulary
tfidf_features_bigram=tfidf_matrix_bigram.toarray()
tfidf_vocab_bigram= tfidf_vectorizer_bigram.get_feature_names_out()

# Print or store TF-IDF features and vocabulary for each vocabulary as needed
print("Bi-gram TF-IDF Features:", tfidf_features_bigram)
print("Bi-gram TF-IDF Vocabulary:", tfidf_vocab_bigram)
```

```
Bi-gram TF-IDF Features: [[0.         0.         0.         ... 0.         0.12900128 0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
```

```
 ...
 [0.         0.         0.25425631 ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
Bi-gram TF-IDF Vocabulary: ['अघ' 'अच' 'अझ' 'अत' 'अध' 'अन' 'अब' 'अभ' 'अर' 'अवस' 'अस' 'आइप' 'आईएमई'
 'आईप' 'आएक' 'आक' 'आकर' 'आज' 'आध' 'आफ' 'आम' 'आय' 'आयल' 'आर' 'आवश' 'इएक'
 'इकर' 'इजर' 'इड' 'इन' 'इनन' 'इनलम' 'इर' 'उक' 'उच' 'उठ' 'उत' 'उन' 'उनक'
 'उनम' 'उनल' 'उपच' 'उम' 'उर' 'उल' 'एउट' 'एक' 'एकअर' 'एकपल' 'एनआईस' 'एभर'
 'एम' 'एल' 'एव' 'एस' 'एसन' 'ऐन' 'ओप' 'कक' 'कथनम' 'कन' 'कब' 'कम' 'कर' 'करण'
 'करणक' 'करणम' 'करल' 'कल' 'कसर' 'कहर' 'खण' 'खर' 'गइरह' 'गक' 'गठन' 'गण'
 'गत' 'गन' 'गम' 'गमक' 'गमल' 'गर' 'घट' 'घटन' 'ङच' 'चतम' 'चन' 'चम' 'चर' 'छन'
 'जक' 'जगढसम' 'जद' 'जन' 'जलव' 'जस' 'ञप' 'टक' 'टन' 'टबल' 'टर' 'टरल' 'टरहर'
 'टव' 'ठक' 'ठकल' 'ठम' 'ठल' 'डक' 'डल' 'ढक' 'णय' 'णयम' 'तथ' 'तन' 'तप' 'तम'
 'तय' 'तर' 'तव' 'थक' 'थनम' 'थप' 'थम' 'दक' 'दन' 'दब' 'दम' 'दर' 'दलक' 'धन'
 'धब' 'धम' 'नक' 'नगर' 'नद' 'नमन' 'नय' 'नल' 'पकमल' 'पछ' 'पत' 'पद' 'पदक'
 'पन' 'पनक' 'पम' 'पर' 'परब' 'पल' 'पह' 'फत' 'बग' 'बज' 'बढ' 'बत' 'बन' 'बर'
 'बरक' 'बल' 'बस' 'भइसक' 'भई' 'भएक' 'भएर' 'भण' 'भद' 'भन' 'मक' 'मट' 'मध'
 'मन' 'मह' 'महत' 'यअन' 'यक' 'यटर' 'यध' 'यन' 'यम' 'यमम' 'यर' 'यरधन' 'यलक'
 'यव' 'यवस' 'यस' 'यसअघ' 'यसक' 'यसब' 'यसम' 'यसल' 'रक' 'रज' 'रण' 'रत' 'रतक'
 'रद' 'रध' 'रब' 'रम' 'रमण' 'रमणक' 'रमम' 'रमश' 'रय' 'रल' 'रव' 'रवक' 'रवर'
 'रष' 'रस' 'रसम' 'रह' 'लक' 'लकहर' 'लखबरम' 'लग' 'लन' 'लम' 'लल' 'लसम' 'वन'
 'वनक' 'वर' 'वरक' 'वरण' 'वरप' 'वल' 'शक' 'शन' 'शनब' 'षक' 'षकसह' 'षड' 'षण'
 'षद' 'षपछ' 'षम' 'षयल' 'षयवस' 'सक' 'सञ' 'सडर' 'सत' 'सध' 'सन' 'सनब' 'सब'
 'सभ' 'सम' 'समर' 'समस' 'सय' 'सरक' 'सल' 'सहकर' 'सहभ' 'हक' 'हज' 'हद' 'हम'
 'हर' '११' '१२' '१४' '१६' '१७३' '१८३' '१९' '१९७१' '२०३९' '२३' '२४' '२५'
 '३०' '३५' '४२' '४४' '५०' '५८' '५९' '६५' '७०' '७२' '७३' '७४' '८१' '८९'
 '९१' '९७७']
```

```
#for trigram
# Create a TfidfVectorizer for tri-grams
tfidf_vectorizer_tri_gram = TfidfVectorizer()

# Fit and transform the documents for one-gram vocabulary
tfidf_matrix_tri_gram = tfidf_vectorizer_tri_gram.fit_transform(df['News'])

# Get the TF-IDF features and vocabulary for one-gram vocabulary
tfidf_features_tri_gram=tfidf_matrix_tri_gram.toarray()
tfidf_vocab_tri_gram= tfidf_vectorizer_tri_gram.get_feature_names_out()

# Print or store TF-IDF features and vocabulary for each vocabulary as needed
print("tri-gram TF-IDF Features:", tfidf_features_tri_gram)
print("tri-gram TF-IDF Vocabulary:", tfidf_vocab_tri_gram)
```

```
tri-gram TF-IDF Features: [[0.         0.         0.         ... 0.         0.12900128 0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.25425631 ... 0.         0.         0.        ]
```

```
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
tri-gram TF-IDF Vocabulary: ['अघ' 'अच' 'अझ' 'अत' 'अध' 'अन' 'अब' 'अभ' 'अर' 'अवस' 'अस' 'आइप' 'आईएमई'
 'आईप' 'आएक' 'आक' 'आकर' 'आज' 'आध' 'आफ' 'आम' 'आय' 'आयल' 'आर' 'आवश' 'इएक'
 'इकर' 'इजर' 'इड' 'इन' 'इनन' 'इनलम' 'इर' 'उक' 'उच' 'उठ' 'उत' 'उन' 'उनक'
 'उनम' 'उनल' 'उपच' 'उम' 'उर' 'उल' 'एउट' 'एक' 'एकअर' 'एकपल' 'एनआईस' 'एभर'
 'एम' 'एल' 'एव' 'एस' 'एसन' 'ऐन' 'ओप' 'कक' 'कथनम' 'कन' 'कब' 'कम' 'कर' 'करण'
 'करणक' 'करणम' 'करल' 'कल' 'कसर' 'कहर' 'खण' 'खर' 'गइरह' 'गक' 'गठन' 'गण'
 'गत' 'गन' 'गम' 'गमक' 'गमल' 'गर' 'घट' 'घटन' 'ङच' 'चतम' 'चन' 'चम' 'चर' 'छन'
 'जक' 'जगढसम' 'जद' 'जन' 'जलव' 'जस' 'ञप' 'टक' 'टन' 'टबल' 'टर' 'टरल' 'टरहर'
 'टव' 'ठक' 'ठकल' 'ठम' 'ठल' 'डक' 'डल' 'ढक' 'णय' 'णयम' 'तथ' 'तन' 'तप' 'तम'
 'तय' 'तर' 'तव' 'थक' 'थनम' 'थप' 'थम' 'दक' 'दन' 'दब' 'दम' 'दर' 'दलक' 'धन'
 'धब' 'धम' 'नक' 'नगर' 'नद' 'नमन' 'नय' 'नल' 'पकमल' 'पछ' 'पत' 'पद' 'पदक'
 'पन' 'पनक' 'पम' 'पर' 'परब' 'पल' 'पह' 'फत' 'बग' 'बज' 'बढ' 'बत' 'बन' 'बर'
 'बरक' 'बल' 'बस' 'भइसक' 'भई' 'भएक' 'भएर' 'भण' 'भद' 'भन' 'मक' 'मट' 'मध'
 'मन' 'मह' 'महत' 'यअन' 'यक' 'यटर' 'यध' 'यन' 'यम' 'यमम' 'यर' 'यरधन' 'यलक'
 'यव' 'यवस' 'यस' 'यसअघ' 'यसक' 'यसब' 'यसम' 'यसल' 'रक' 'रज' 'रण' 'रत' 'रतक'
 'रद' 'रध' 'रब' 'रम' 'रमण' 'रमणक' 'रमम' 'रमश' 'रय' 'रल' 'रव' 'रवक' 'रवर'
 'रष' 'रस' 'रसम' 'रह' 'लक' 'लकहर' 'लखबरम' 'लग' 'लन' 'लम' 'लल' 'लसम' 'वन'
 'वनक' 'वर' 'वरक' 'वरण' 'वरप' 'वल' 'शक' 'शन' 'शनब' 'षक' 'षकसह' 'षड' 'षण'
 'षद' 'षपछ' 'षम' 'षयल' 'षयवस' 'सक' 'सञ' 'सडर' 'सत' 'सध' 'सन' 'सनब' 'सब'
 'सभ' 'सम' 'समर' 'समस' 'सय' 'सरक' 'सल' 'सहकर' 'सहभ' 'हक' 'हज' 'हद' 'हम'
 'हर' '११' '१२' '१४' '१६' '१७३' '१८३' '१९' '१९७१' '२०३९' '२३' '२४' '२५'
 '३०' '३५' '४२' '४४' '५०' '५८' '५९' '६५' '७०' '७२' '७३' '७४' '८१' '८९'
 '९१' '९७७']
```

```
done('Task 3')
```

```
==================================
Task 3: 2023-09-25 14:41:45.105462
==================================
```

Task 4: Compute document similarity matrix (if your document size = N , this will result in the NxN matrix) for each vocab list.

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer


#4) Task 4: Compute document similarity matrix (if your document size = N , this will result in the NxN matrix) for each vocal list.
from sklearn.metrics.pairwise import cosine_similarity

# Assuming you have already created tfidf_matrix_onegram, tfidf_matrix_bigram, and tfidf_matrix_trigram

# Compute cosine similarity matrices
cosine_similarity_onegram = cosine_similarity(tfidf_matrix_onegram, tfidf_matrix_onegram)
cosine_similarity_bigram = cosine_similarity(tfidf_matrix_bigram, tfidf_matrix_bigram)
```

```
cosine_similarity_tri_gram = cosine_similarity(tfidf_matrix_tri_gram, tfidf_matrix_tri_gram)

# cosine_similarity_onegram, cosine_similarity_bigram, and cosine_similarity_trigram are NxN matrices
print(cosine_similarity_onegram)

print(cosine_similarity_tri_gram)
```

```
 [0.04602077 0.01301387 0.06516523 0.02917168 0.0208128  0.01783019
  0.09796296 0.05899303 0.03188343 0.05482784 0.07668687 0.0731918
  0.04653723 0.04536175 0.13969809 0.00551079 1.         0.17512596
  0.15984821 0.03501628]
 [0.02282665 0.01806232 0.10203738 0.         0.01590768 0.02865185
  0.02968965 0.01648659 0.07333116 0.0431229  0.0238767  0.08143527
  0.05123473 0.02099903 0.0443889  0.         0.17512596 1.
  0.10242776 0.04903338]
 [0.05822291 0.03375776 0.03244675 0.03040233 0.03542937 0.
  0.03193383 0.01773278 0.02277254 0.         0.         0.
  0.0515554  0.0540347  0.         0.         0.15984821 0.10242776
  1.         0.07207038]
 [0.13022456 0.09530865 0.06535484 0.03194314 0.00671426 0.02089029
  0.01456456 0.07054685 0.03925735 0.0642871  0.07330619 0.01240168
  0.17724069 0.12394578 0.01078231 0.05982144 0.03501628 0.04903338
  0.07207038 1.        ]]
```

```
print(cosine_similarity_bigram)
```

```
  0.13841256 0.06604313 0.09262145 0.03314811 0.01783019 0.02865185
  0.         0.02089029]
 [0.06546364 0.06404776 0.05785574 0.04425207 0.0225581  0.12128221
  1.         0.11818785 0.06122687 0.26605095 0.2516376  0.10228419
```

```
 [0.14616162 0.23852946 0.11166132 0.03722063 0.16183428 0.13841256
  0.09033447 0.06122629 0.11111207 0.24558846 0.13370784 0.14261403
  1.         0.17886626 0.14818937 0.1565588  0.04653723 0.05123473
  0.0515554  0.17724069]
 [0.13743434 0.10513605 0.11125477 0.07187011 0.04493125 0.06604313
  0.08642472 0.11246811 0.11543581 0.17664869 0.13782178 0.06113938
  0.17886626 1.         0.07779924 0.14571504 0.04536175 0.02099903
  0.0540347  0.12394578]
 [0.03939811 0.02020173 0.02268732 0.05011707 0.0925144  0.09262145
  0.12168914 0.04567545 0.20730772 0.20885878 0.11362992 0.06951952
  0.14818937 0.07779924 1.         0.0342319  0.13969809 0.0443889
  0.         0.01078231]
 [0.11102509 0.1696203  0.08552682 0.16861782 0.09672058 0.03314811
  0.0320751  0.04380598 0.05413193 0.07234296 0.2632783  0.10080796
  0.1565588  0.14571504 0.0342319  1.         0.00551079 0.
  0.         0.05982144]
 [0.04602077 0.01301387 0.06516523 0.02917168 0.0208128  0.01783019
  0.09796296 0.05899303 0.03188343 0.05482784 0.07668687 0.0731918
  0.04653723 0.04536175 0.13969809 0.00551079 1.         0.17512596
  0.15984821 0.03501628]
 [0.02282665 0.01806232 0.10203738 0.         0.01590768 0.02865185
  0.02968965 0.01648659 0.07333116 0.0431229  0.0238767  0.08143527
  0.05123473 0.02099903 0.0443889  0.         0.17512596 1.
  0.10242776 0.04903338]
 [0.05822291 0.03375776 0.03244675 0.03040233 0.03542937 0.
  0.03193383 0.01773278 0.02277254 0.         0.         0.
  0.0515554  0.0540347  0.         0.         0.15984821 0.10242776
  1.         0.07207038]
 [0.13022456 0.09530865 0.06535484 0.03194314 0.00671426 0.02089029
  0.01456456 0.07054685 0.03925735 0.0642871  0.07330619 0.01240168
  0.17724069 0.12394578 0.01078231 0.05982144 0.03501628 0.04903338
  0.07207038 1.        ]]
```

```
print(cosine_similarity_tri_gram)
```

```
[[1.         0.47395226 0.04127486 0.04214786 0.0789131  0.064146
  0.06546364 0.09467856 0.05562955 0.11750804 0.0966366  0.06767683
  0.14616162 0.13743434 0.03939811 0.11102509 0.04602077 0.02282665
  0.05822291 0.13022456]
 [0.47395226 1.         0.05073631 0.07421978 0.10135262 0.01957002
  0.06404776 0.08793772 0.08141335 0.12203887 0.20934162 0.23869811
  0.23852946 0.10513605 0.02020173 0.1696203  0.01301387 0.01806232
  0.03375776 0.09530865]
 [0.04127486 0.05073631 1.         0.06124683 0.03055637 0.11620479
  0.05785574 0.06703448 0.07817903 0.08948571 0.12008332 0.07161207
  0.11166132 0.11125477 0.02268732 0.08552682 0.06516523 0.10203738
```

```
  0.03244675 0.06535484]
 [0.04214786 0.07421978 0.06124683 1.         0.06536618 0.02372947
  0.04425207 0.02115952 0.05285832 0.08633517 0.09758023 0.02433462
  0.03722063 0.07187011 0.05011707 0.16861782 0.02917168 0.
  0.03040233 0.03194314]
 [0.0789131  0.10135262 0.03055637 0.06536618 1.         0.05530631
  0.0225581  0.09384142 0.04836971 0.12396269 0.1228423  0.1554968
  0.16183428 0.04493125 0.0925144  0.09672058 0.0208128  0.01590768
  0.03542937 0.00671426]
 [0.064146   0.01957002 0.11620479 0.02372947 0.05530631 1.
  0.12128221 0.06974471 0.08796578 0.09463654 0.08259523 0.06288706
  0.13841256 0.06604313 0.09262145 0.03314811 0.01783019 0.02865185
  0.         0.02089029]
 [0.06546364 0.06404776 0.05785574 0.04425207 0.0225581  0.12128221
  1.         0.11818785 0.06122687 0.26605095 0.2516376  0.10228419
  0.09033447 0.08642472 0.12168914 0.0320751  0.09796296 0.02968965
  0.03193383 0.01456456]
 [0.09467856 0.08793772 0.06703448 0.02115952 0.09384142 0.06974471
  0.11818785 1.         0.08277746 0.13909179 0.13646806 0.14570318
  0.06122629 0.11246811 0.04567545 0.04380598 0.05899303 0.01648659
  0.01773278 0.07054685]
 [0.05562955 0.08141335 0.07817903 0.05285832 0.04836971 0.08796578
  0.06122687 0.08277746 1.         0.1859378  0.05926284 0.04099602
  0.11111207 0.11543581 0.20730772 0.05413193 0.03188343 0.07333116
  0.02277254 0.03925735]
 [0.11750804 0.12203887 0.08948571 0.08633517 0.12396269 0.09463654
  0.26605095 0.13909179 0.1859378  1.         0.33937928 0.11497435
  0.24558846 0.17664869 0.20885878 0.07234296 0.05482784 0.0431229
  0.         0.0642871 ]
 [0.0966366  0.20934162 0.12008332 0.09758023 0.1228423  0.08259523
  0.2516376  0.13646806 0.05926284 0.33937928 1.         0.29348558
  0.13370784 0.13782178 0.11362992 0.2632783  0.07668687 0.0238767
  0.         0.07330619]
 [0.06767683 0.23869811 0.07161207 0.02433462 0.1554968  0.06288706
  0.10228419 0.14570318 0.04099602 0.11497435 0.29348558 1.
  0.14261403 0.06113938 0.06951952 0.10080796 0.0731918  0.08143527
  0.         0.01240168]
 [0.14616162 0.23852946 0.11166132 0.03722063 0.16183428 0.13841256
  0.09033447 0.06122629 0.11111207 0.24558846 0.13370784 0.14261403
  1.         0.17886626 0.14818937 0.1565588  0.04653723 0.05123473
  0.0515554  0.17724069]
 [0.13743434 0.10513605 0.11125477 0.07187011 0.04493125 0.06604313
  0.08642472 0.11246811 0.11543581 0.17664869 0.13782178 0.06113938
  0.17886626 1.         0.07779924 0.14571504 0.04536175 0.02099903
  0.0540347  0.12394578]
 [0.03939811 0.02020173 0.02268732 0.05011707 0.0925144  0.09262145
  0.12168914 0.04567545 0.20730772 0.20885878 0.11362992 0.06951952
```

```
done('Task 4')
```

```
==================================
Task 4: 2023-09-25 14:42:00.310593
==================================
```

Task 5: Write your interpretation on the result of Task 4.

```
#Task 5: Write your interpretation on the result of Task 4.



# We found the similarity between words in documents.


done('Task 5')
```

```
==================================
Task 5: 2023-09-25 14:50:33.475915
==================================
```