

**NLP ASSIGNMENT 2**

**SUBMITTED BY: AHMER KHAN**

**SUBMITTED TO: SIR ABDUL AHAD**

**ROLL NO: 10203**

**DATE: 30/10/2024**

**BSSE-7<sup>TH</sup> A**

## QUESTION NO 1

Choose a dataset of your choice and apply preprocessing steps on it as was discussed in class. i.e Text Lowercase, Tokenization, Removal of (Punctuations, stop words, numeric tokens, html/xml tags, white spaces ), stemming and Lemmatization, spell correction, handling special characters etc.

**DATASET LINK:** [Indian Migration History \(kaggle.com\)](https://www.kaggle.com/datasets/indian-migration-history)

### SCREENSHOTS OF ALL PREPROCESSING:

The screenshot shows a Jupyter Notebook interface with the following content:

```
[1] import pandas as pd
```

```
[2] data=pd.read_csv('/content/drive/MyDrive/datasets/IndianMigrationHistory1.3.csv')
```

```
data.head
```

	Country Origin Name	Country Origin Code	Migration by Sender Name
0	India	IND	Female
1	India	IND	Female
2	India	IND	Female
3	India	IND	Female
4	India	IND	Female
...	...	...	...
457	India	IND	Male
458	India	IND	Male
459	India	IND	Male
460	India	IND	Male
461	India	IND	Male

	Migration by Gender Code	Country Dest Name	Country Dest Code
0	FEM	Afghanistan	AFG
1	FEM	Albania	ALB
2	FEM	Algeria	DZA
3	FEM	American Samoa	ASM
4	FEM	Andorra	AND

2s completed at 8:29 PM

Colab Research Interface showing a Jupyter Notebook titled "Untitled3.ipynb". The notebook is open to the "Code" tab. The file explorer on the left shows a directory structure with "drive", "MyDrive", and "sample\_data".

The code cell [6] contains the following Python code:

```
[6] dataset_path = '/content/drive/MyDrive/datasets/IndianMigrationHistory1.3.csv'
df = pd.read_csv(dataset_path)
```

The output of the code cell [7] shows the columns of the DataFrame:

```
[7] df.columns
Index(['Country Origin Name', 'Country Origin Code',
      'Migration by Gender Name', 'Migration by Gender Code',
      'Country Dest Name', 'Country Dest Code', '1968 [1968]', '1978 [1978]',
      '1988 [1988]', '1998 [1998]', '2008 [2008]'],
      dtype='object')
```

The output also displays a preview of the data:

	1968 [1968]	1978 [1978]	1988 [1988]	1998 [1998]	2008 [2008]
0	8521	11578	2072	2234	4445
1	5	2	2	2	2
2	6	2	2	1	8
3	8	8	2	3	4
4	2	13	3	6	8
...	...	...	...	...	...
457	8	8	8	8	2
458	74	89	185	138	213
459	3171	3026	4868	6617	7834
460	4697	3746	3599	2695	2414
461	1496	1485	1325	1258	1149

The output concludes with: [462 rows x 11 columns]>

The bottom status bar indicates the notebook is "Partly cloudy" and the system clock shows 8:42 PM on 10/24/2023.

Colab Research Interface showing a Jupyter Notebook titled "Untitled3.ipynb". The notebook is open to the "Code" tab. The file explorer on the left shows a directory structure with "drive", "MyDrive", and "sample\_data".

The code cell [7] contains the following Python code:

```
[7] df.columns
Index(['Country Origin Name', 'Country Origin Code',
      'Migration by Gender Name', 'Migration by Gender Code',
      'Country Dest Name', 'Country Dest Code', '1968 [1968]', '1978 [1978]',
      '1988 [1988]', '1998 [1998]', '2008 [2008]'],
      dtype='object')
```

The code cell [8] contains the following Python code:

```
[8] !pip install autocorrect
```

The output of the code cell [8] shows the installation process:

```
Collecting autocorrect
  Downloading autocorrect-2.6.1.tar.gz (622 kB)
    Preparing metadata (setup.py) ... done
Building wheels for collected packages: autocorrect
  Building wheel for autocorrect (setup.py) ... done
  Created wheel for autocorrect: filename=autocorrect-2.6.1-py3-none-any.whl size=622181 sha256=4138ad291533c9d8031
  Stored in directory: /root/.cache/pip/wheels/b5/7b/6d/b76b29ca11ff8e2521c8c7d68e5bfee4f81789d761931243d3
Successfully built autocorrect
Installing collected packages: autocorrect
Successfully installed autocorrect-2.6.1
```

The code cell [9] contains the following Python code:

```
[9] import pandas as pd
import nltk
from nltk.corpus import stopwords
```

The bottom status bar indicates the notebook is "Partly cloudy" and the system clock shows 8:42 PM on 10/24/2023.

The screenshot shows a Google Colab notebook interface. The top bar includes a menu (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar indicating 'All changes saved'. The left sidebar shows a file explorer with a 'drive' folder containing 'MyDrive' and 'sample\_data'. The main code area contains the following code:

```
Successfully installed autocorrect-2.6.1

import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
import re
from bs4 import BeautifulSoup
import string
from autocorrect import Speller

[11] import nltk
nltk.download('wordnet')

[nltk_data] Downloading package wordnet to /root/nltk_data...
True

[12] # Sample DataFrame
data = {
    'Country Origin Name': ['India', 'India', 'India', 'India', 'India'],
    'Country Dest Name': ['Afghanistan', 'Albania', 'Algeria', 'American Samoa', 'Andorra']
}
```

The bottom status bar shows '2s completed at 8:29 PM'.

The screenshot shows the same Google Colab notebook, now displaying the continuation of the code from the previous state. The code defines preprocessing functions and creates a DataFrame:

```
[12] # Sample DataFrame
data = {
    'Country Origin Name': ['India', 'India', 'India', 'India', 'India'],
    'Country Dest Name': ['Afghanistan', 'Albania', 'Algeria', 'American Samoa', 'Andorra']
}

df = pd.DataFrame(data)

# Define preprocessing functions
nltk.download('punkt')
nltk.download('stopwords')

stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
spell = Speller()

def preprocess_text(text):
    # Lowercase
    text = text.lower()

    # Tokenization
    words = word_tokenize(text)

    # Remove punctuation, numbers, and HTML/XML tags
```

The bottom status bar shows '2s completed at 8:29 PM'.

```
Untitled3.ipynb
File Edit View Insert Runtime Tools Help All changes saved

[12]
# Tokenization
words = word_tokenize(text)

# Remove punctuation, numbers, and HTML/XML tags
words = [re.sub(r'[{}]' .format(string.punctuation), '', word) for word in words]
words = [word for word in words if not word.isnumeric()]
words = [re.sub(r'<.*>', '', word) for word in words]

# Remove stop words
words = [word for word in words if word not in stop_words]

# Stemming
words = [stemmer.stem(word) for word in words]

# Lemmatization
words = [lemmatizer.lemmatize(word) for word in words]

# Spell correction
words = [spell(word) for word in words]

return ' '.join(words)

# Apply preprocessing to text columns
text_columns = ['Country Origin Name', 'Country Dest Name']
for col in text_columns:
    df[col] = df[col].apply(preprocess_text)
    ✓ 2s completed at 8:29 PM
```

```
Untitled3.ipynb
File Edit View Insert Runtime Tools Help All changes saved

[12] text_columns = ['Country Origin Name', 'Country Dest Name']
for col in text_columns:
    df[col] = df[col].apply(preprocess_text)

print(df)

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
Country Origin Name Country Dest Name
0 India afghanistan
1 India albania
2 India algeria
3 India american samoa
4 India pandora

[]
```

## QUESTION NO 2

Write a program in python which utilizes regular expression (R.E) to perform the following tasks.

Email Address Extraction

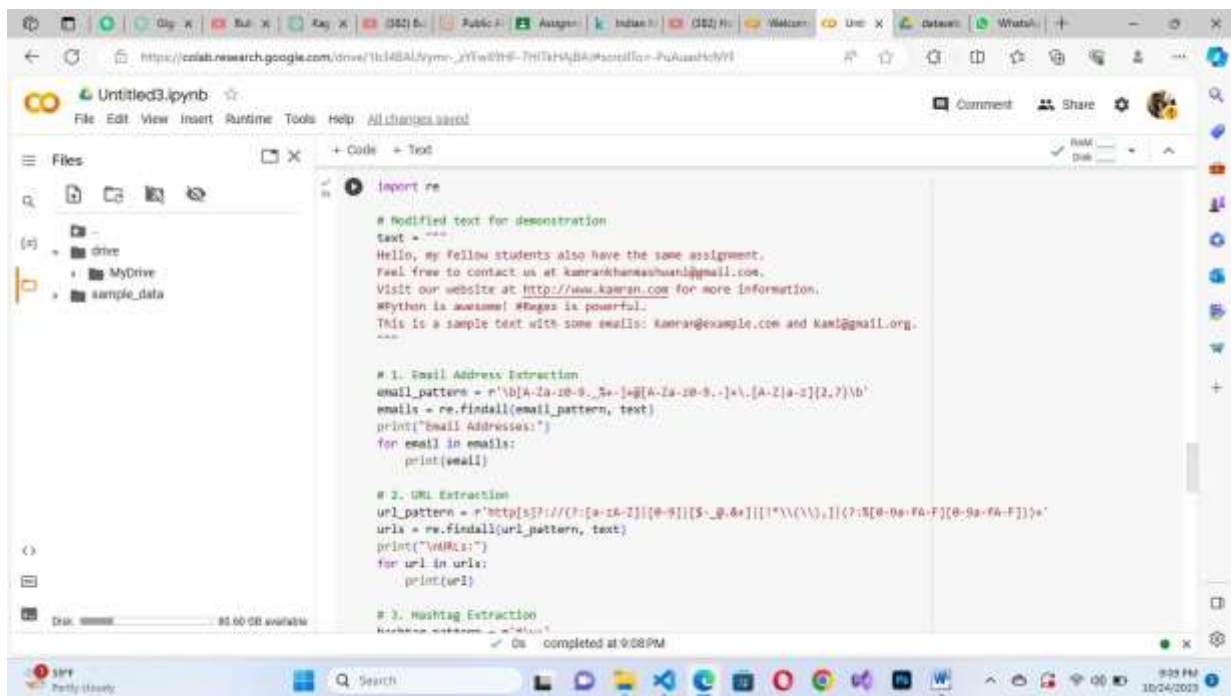
URL Extraction

Hashtag Extraction

Sentence Segmentation

Search and replace with capture groups. (search and replace certain text.)

## SOLUTION:



```
import re

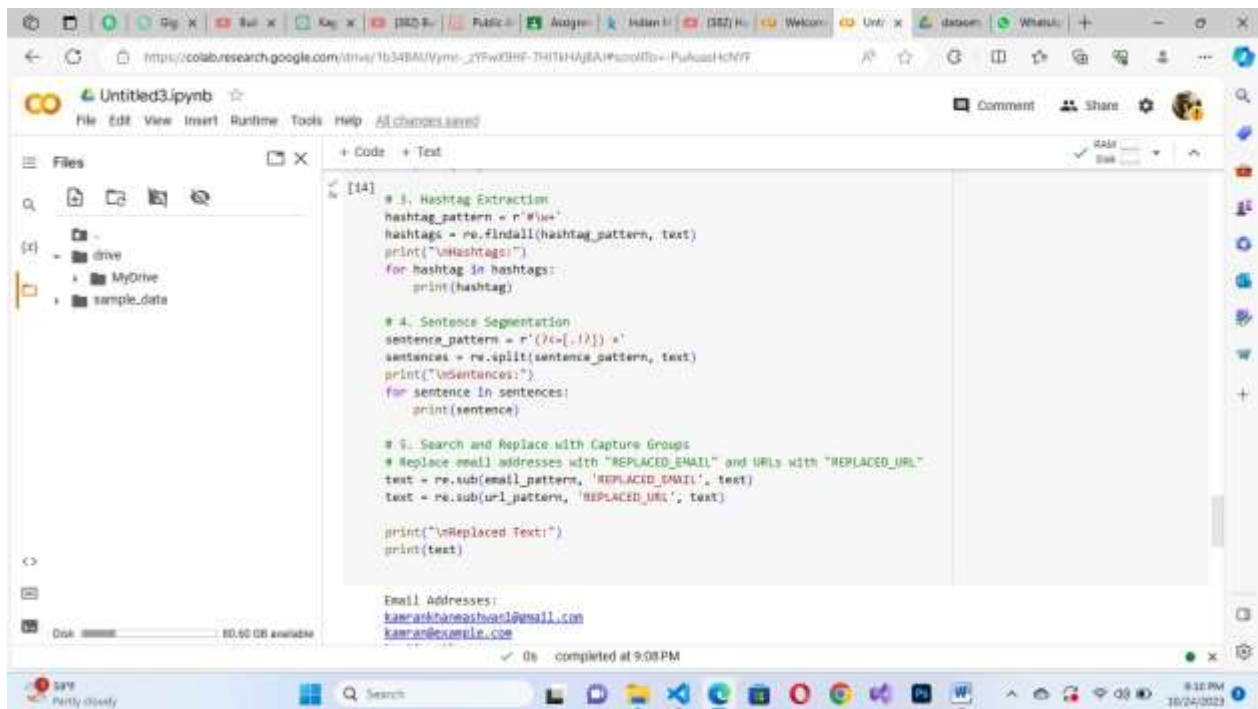
# Modified text for demonstration
text = """
Hello, my fellow students also have the same assignment.
Feel free to contact us at kamranhussain@gmail.com.
Visit our website at http://www.kamran.com for more information.
Python is awesome! #Regex is powerful.
This is a sample text with some emails: kamran@example.com and kam@gmail.org.
"""

# 1. Email Address Extraction
email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,7}\b'
emails = re.findall(email_pattern, text)
print("Email Addresses:")
for email in emails:
    print(email)

# 2. URL Extraction
url_pattern = r'http[s]?://(?:[a-zA-Z]{0-9}+|[\w-]+)|[0-9]{1-3}(\.[0-9]{1-3}){0-3}|[0-9]{1-3}(\.[0-9]{1-3}){0-3}'
urls = re.findall(url_pattern, text)
print("URLs:")
for url in urls:
    print(url)

# 3. Hashtag Extraction
hashtag_pattern = r'#\w{1,15}'
hashtags = re.findall(hashtag_pattern, text)
print("Hashtags:")
for hashtag in hashtags:
    print(hashtag)
```





The screenshot shows a Google Colab notebook titled 'Untitled3.ipynb'. The left sidebar displays a file explorer with folders named 'drive', 'MyDrive', and 'sample\_data'. The main code area contains the following Python code:

```
[141] # 3. Hashtag Extraction
hashtag_pattern = r'#\w+'
hashtags = re.findall(hashtag_pattern, text)
print("\nHashtags:")
for hashtag in hashtags:
    print(hashtag)

# 4. Sentence Segmentation
sentence_pattern = r'(?<[.!?]) +'
sentences = re.split(sentence_pattern, text)
print("\nsentences:")
for sentence in sentences:
    print(sentence)

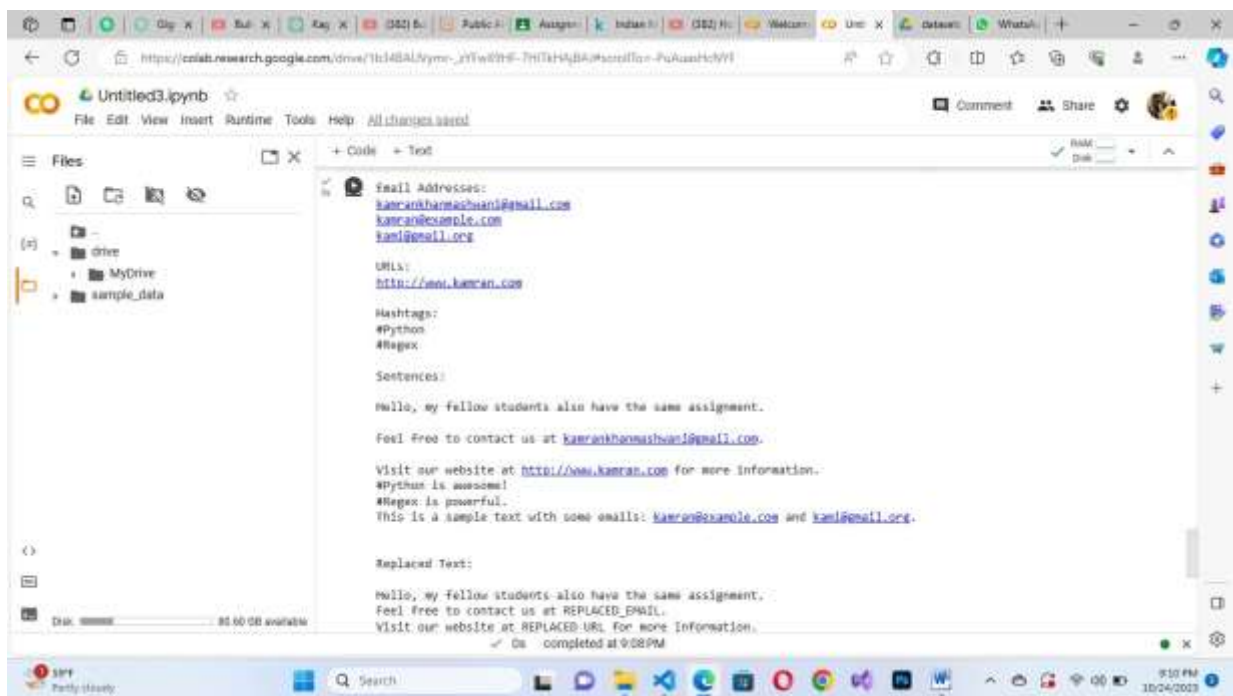
# 5. Search and Replace with Capture Groups
# Replace email addresses with "REPLACED_EMAIL" and URLs with "REPLACED_URL"
text = re.sub(email_pattern, 'REPLACED_EMAIL', text)
text = re.sub(url_pattern, 'REPLACED_URL', text)

print("\nReplaced Text:")
print(text)
```

Below the code, the output shows the extracted email addresses:

```
Email Addresses:
kamaranhamashwan@gmail.com
kamaran@example.com
```

The bottom status bar indicates the execution is completed at 9:08 PM.



This screenshot shows the same Google Colab notebook after execution. The output is displayed in a single cell, showing the results of the text processing:

```
Email Addresses:
kamaranhamashwan@gmail.com
kamaran@example.com
kam@gmail.org

URLs:
http://www.kamran.com

Hashtags:
#Python
#Regex

Sentences:

Hello, my fellow students also have the same assignment.

Feel free to contact us at kamaranhamashwan@gmail.com.

Visit our website at http://www.kamran.com for more information.
#Python is awesome!
#Regex is powerful.
This is a sample text with some emails: kamaran@example.com and kam@gmail.org.

Replaced Text:

Hello, my fellow students also have the same assignment.
Feel free to contact us at REPLACED_EMAIL.
Visit our website at REPLACED_URL for more information.
```

The bottom status bar shows the execution is completed at 9:08 PM.

Untitled3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- drive
- MyDrive
- sample\_data

Code

```
[14]: Hashtags:
#Python
#Regex

Sentences:

Hello, my fellow students also have the same assignment.

Feel free to contact us at kaaranhgonashwan@gmail.com.

Visit our website at http://www.kaaran.com for more information.
#Python is awesome!
#Regex is powerful.
This is a sample text with some emails: kaaran@sample.com and kaal@gmail.org.

Replaced text:

Hello, my fellow students also have the same assignment.
Feel free to contact us at REPLACED_EMAIL.
Visit our website at REPLACED_URL for more information.
#Python is awesome! #Regex is powerful.
This is a sample text with some emails: REPLACED_EMAIL and REPLACED_EMAIL.
```

81.60 GB available

completed at 9:08 PM

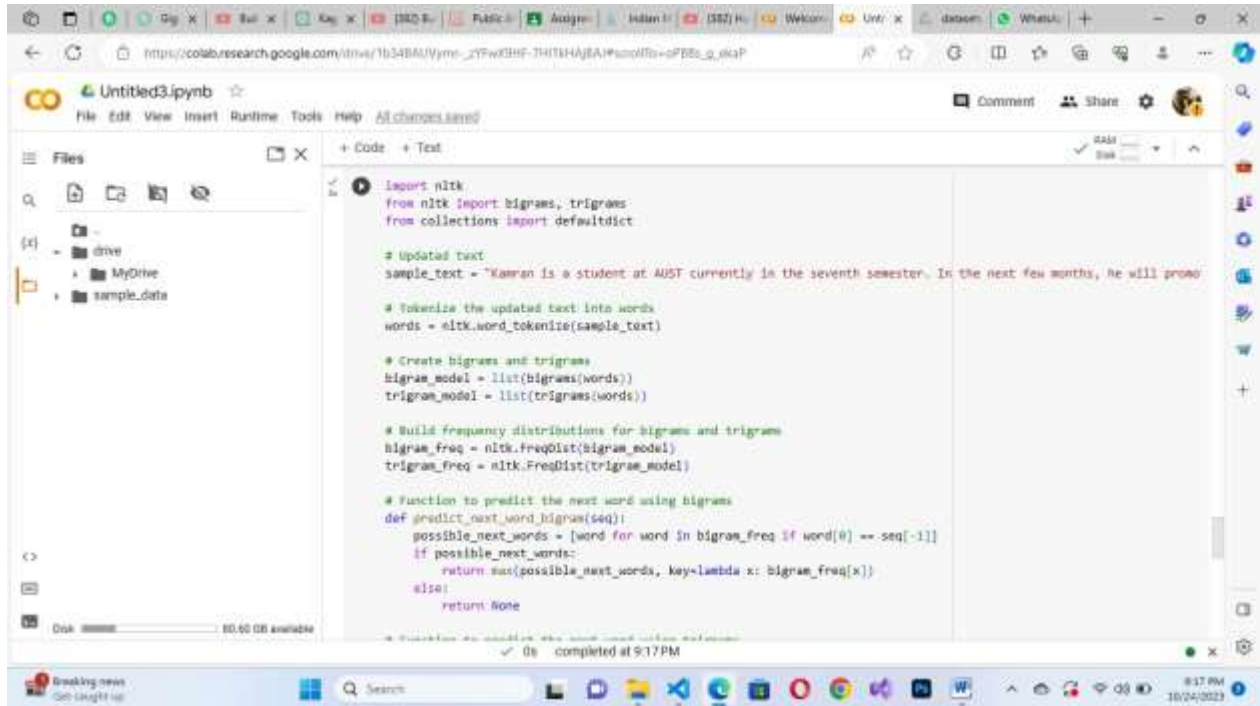
9:10 PM 10/04/2023



## QUESTION 3

Text Prediction using N-Gram models. Given a sequence of words, predict the next word using bigrams or trigrams.

### SOLUTION:



```
import nltk
from nltk import bigrams, trigrams
from collections import defaultdict

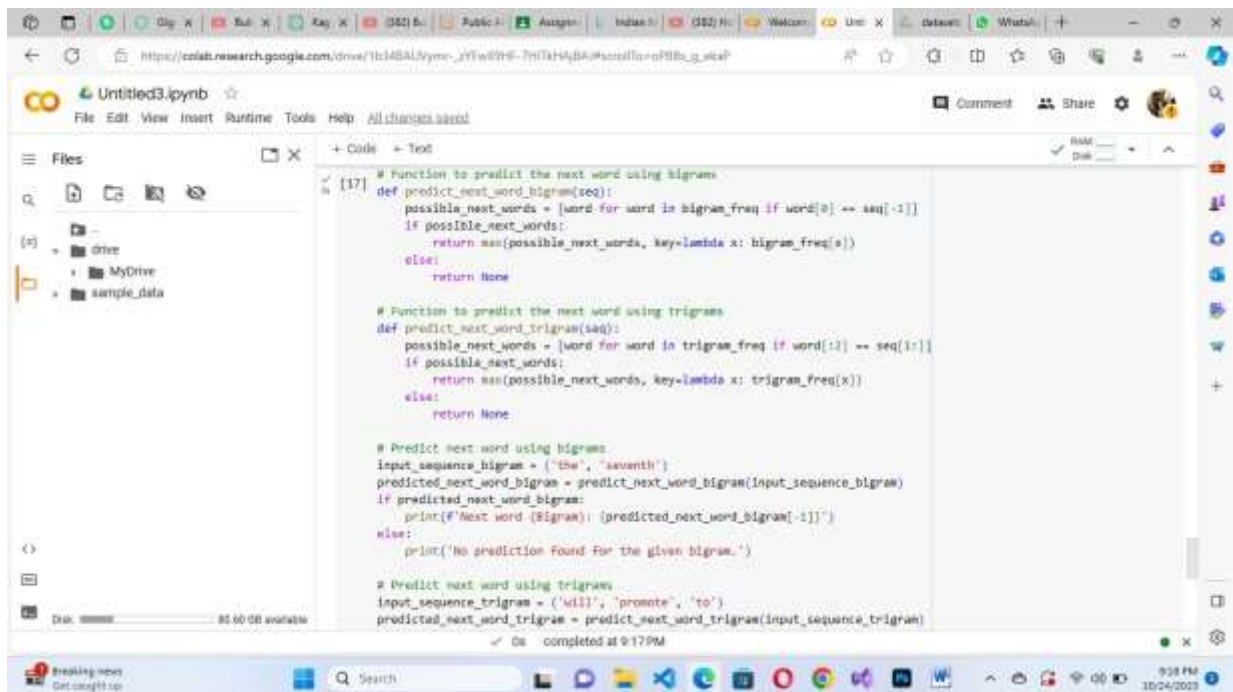
# Updated text
sample_text = "Kamran is a student at AUST currently in the seventh semester. In the next few months, he will prom"

# Tokenize the updated text into words
words = nltk.word_tokenize(sample_text)

# Create bigrams and trigrams
bigram_model = list(bigrams(words))
trigram_model = list(trigrams(words))

# Build frequency distributions for bigrams and trigrams
bigram_freq = nltk.FreqDist(bigram_model)
trigram_freq = nltk.FreqDist(trigram_model)

# Function to predict the next word using bigrams
def predict_next_word_bigram(seq):
    possible_next_words = [word for word in bigram_freq if word[0] == seq[-1]]
    if possible_next_words:
        return max(possible_next_words, key=lambda x: bigram_freq[x])
    else:
        return None
```



```
# Function to predict the next word using bigrams
def predict_next_word_bigram(seq):
    possible_next_words = [word for word in bigram_freq if word[0] == seq[-1]]
    if possible_next_words:
        return max(possible_next_words, key=lambda x: bigram_freq[x])
    else:
        return None

# Function to predict the next word using trigrams
def predict_next_word_trigram(seq):
    possible_next_words = [word for word in trigram_freq if word[-1] == seq[-1]]
    if possible_next_words:
        return max(possible_next_words, key=lambda x: trigram_freq[x])
    else:
        return None

# Predict next word using bigrams
input_sequence_bigram = ("the", "seventh")
predicted_next_word_bigram = predict_next_word_bigram(input_sequence_bigram)
if predicted_next_word_bigram:
    print(f'Next word (Bigram): {predicted_next_word_bigram[-1]}')
else:
    print('No prediction found for the given bigram.')

# Predict next word using trigrams
input_sequence_trigram = ("will", "promote", "to")
predicted_next_word_trigram = predict_next_word_trigram(input_sequence_trigram)
```

Untitled3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- drive
- MyDrive
- sample\_data

```
# Predict next word using bigrams
input_sequence_bigram = ('the', 'seventh')
predicted_next_word_bigram = predict_next_word_bigram(input_sequence_bigram)
if predicted_next_word_bigram:
    print(f'Next word (Bigram): {predicted_next_word_bigram[-1]}')
else:
    print('No prediction found for the given bigram.')

# Predict next word using trigrams
input_sequence_trigram = ('will', 'promote', 'to')
predicted_next_word_trigram = predict_next_word_trigram(input_sequence_trigram)
if predicted_next_word_trigram:
    print(f'Next word (Trigram): {predicted_next_word_trigram[-1]}')
else:
    print('No prediction found for the given trigram.')

Next word (Bigram): semester
Next word (Trigram): the
```

completed at 9:17 PM

9:18 PM 10/04/2023