ASSIGNMENT- 4

(21BCS2049, NIKHIL KUMAR CHAHAR)

PART 1:-

WEB SCRAPING:-

```
import requests
from bs4 import BeautifulSoup

def scrape_articles(url):
    # Send a GET request to the URL
    response = requests.get(url)

    # Check if the request was successful
    if response.status_code != 200:
        print(f"Failed to retrieve the webpage. Status code:
{response.status_code}")
        return []

    # Parse the webpage content
    soup = BeautifulSoup(response.content, 'html.parser')

    # Find all article elements
    articles = []
    for item in soup.find_all('article'):
        # Extract the title
        title_tag = item.find('h2')
        if title_tag:
            title = title_tag.get_text()
        else:
            continue

        # Extract the link
        link_tag = item.find('a', href=True)
        if link_tag:
            link = link_tag['href']
        else:
            continue

        articles.append({'title': title, 'link': link})

    return articles

def save_to_text_file(articles, filename):
    full_path = r'C:\Users\nikhilchahar\Desktop\NLP CLASS\\' + filename
    with open(full_path, 'w', encoding='utf-8') as f:
        for article in articles:
            f.write(f"Title: {article['title']}\n")
```

```
            f.write(f"Link: {article['link']}\n")
            f.write("-" * 80 + "\n")

if __name__ == "__main__":
    url = 'https://www.bbc.co.uk/bitesize/articles/zhtqcqt
    articles = scrape_articles(url)

    # Save the scraped articles to a text file in the specified directory
    save_to_text_file(articles, 'assignment_5_data.txt')

    print(f"Saved {len(articles)} articles to
C:\\Users\\nikhilchahar\\Desktop\\NLP CLASS\\assignment_4_data.txt")
```

OUTPUT:-

```
Saved 32 articles to C:\Users\nikhilchahar\Desktop\NLP CLASS\assignment_5_data.txt
PS C:\Users\nikhilchahar>
```

PART 2:-

TRANSLATING FROM ENGLISH TO HINDI

```
from deep_translator import GoogleTranslator

def translate_and_copy_file_content(input_file, output_file, char_limit=4000):
    # Read input file
    with open(input_file, 'r', encoding='utf-8') as f:
        content = f.read()

    # Take only the first 4000 characters
    content_to_translate = content[:char_limit]

    try:
        # Translate content to Hindi
        translated_text = GoogleTranslator(source='auto',
target='hi').translate(content_to_translate)

        # Write translated content to output file
        with open(output_file, 'w', encoding='utf-8') as f:
            f.write(translated_text)

        print(f"Content copied from {input_file} and translated to Hindi in
{output_file}")

    except Exception as e:
```

```
        print(f"An error occurred during translation: {e}")

if __name__ == "__main__":
    # Define file paths
    input_file = r'C:\Users\nikhilchahar\Desktop\NLP
CLASS\assignment_4_data.txt'
    output_file = r'C:\Users\nikhilchahar\Desktop\NLP
CLASS\assignment_4_hindi.txt'

    # Translate content from input file and write to output file
    translate_and_copy_file_content(input_file, output_file)
from deep_translator import GoogleTranslator

def translate_and_copy_file_content(input_file, output_file, char_limit=4000):
    # Read input file
    with open(input_file, 'r', encoding='utf-8') as f:
        content = f.read()

    # Take only the first 4000 characters
    content_to_translate = content[:char_limit]

    try:
        # Translate content to Hindi
        translated_text = GoogleTranslator(source='auto',
target='hi').translate(content_to_translate)

        # Write translated content to output file
        with open(output_file, 'w', encoding='utf-8') as f:
            f.write(translated_text)

        print(f"Content copied from {input_file} and translated to Hindi in
{output_file}")

    except Exception as e:
        print(f"An error occurred during translation: {e}")

if __name__ == "__main__":
    # Define file paths
    input_file = r'C:\Users\nikhilchahar\Desktop\NLP
CLASS\assignment_5_data.txt'
    output_file = r'C:\Users\nikhilchahar\Desktop\NLP
CLASS\assignment_5_data_hindi.txt'

    # Translate content from input file and write to output file
    translate_and_copy_file_content(input_file, output_file)
```

OUTPUT:-

```
Content copied from C:\Users\nikhilchahar\Desktop\NLP CLASS\assignment_4_data.txt and translated to Hindi in C:\Users\nikhilchahar\Desktop\NLP CLASS\assignment_4_hindi.txt
Content copied from C:\Users\nikhilchahar\Desktop\NLP CLASS\assignment_5_data.txt and translated to Hindi in C:\Users\nikhilchahar\Desktop\NLP CLASS\assignment_5_data_hindi.txt
```

PART 3:-

EVALUATING DATA

```
from deep_translator import GoogleTranslator
from nltk.translate.bleu_score import sentence_bleu
import nltk

# Ensure NLTK resources are downloaded
nltk.download('punkt')

def translate_content(content, char_limit=4000):
    # Take only the first 4000 characters
    content_to_translate = content[:char_limit]

    try:
        # Translate content to Hindi
        translated_text = GoogleTranslator(source='auto',
target='hi').translate(content_to_translate)
        return translated_text
    except Exception as e:
        print(f"An error occurred during translation: {e}")
        return None

def evaluate_translation(translated_text, reference_text):
    # Tokenize the sentences
    translated_sentences = nltk.sent_tokenize(translated_text)
    reference_sentences = nltk.sent_tokenize(reference_text)

    # Tokenize the words
    translated_tokens = [nltk.word_tokenize(sent) for sent in
translated_sentences]
    reference_tokens = [[nltk.word_tokenize(sent)] for sent in
reference_sentences]

    # Calculate BLEU score
    scores = []
    for trans, ref in zip(translated_tokens, reference_tokens):
        score = sentence_bleu(ref, trans)
        scores.append(score)

    average_bleu_score = sum(scores) / len(scores)
    return average_bleu_score
```

```
if __name__ == "__main__":
    # Define file paths
    input_file = r'C:\Users\nikhilchahar\Desktop\NLP
CLASS\assignment_5_data.txt'
    reference_file = r"C:\Users\nikhilchahar\Desktop\NLP
CLASS\assignment_5_data_hindi.txt"

    # Read input file
    with open(input_file, 'r', encoding='utf-8') as f:
        content = f.read()

    # Translate content
    translated_text = translate_content(content)

    if translated_text:
        # Read reference file
        with open(reference_file, 'r', encoding='utf-8') as f:
            reference_text = f.read()

        # Evaluate translation
        bleu_score = evaluate_translation(translated_text, reference_text)
        print(f"Average BLEU score for the translated content: {bleu_score}")
    else:
        print("Translation failed, skipping evaluation.")
```

OUTPUT:-

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\nikhilchahar\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Average BLEU score for the translated content: 1.0
```