Documentation Assignment 1: Natural Language Processing

Team Members:

Nimitt

Pratham Sharda

Pranjal Gaur

Shubham Aggarwal

Harshit

Chirag Patel

Code Files Repo :

The following repo contains the base code that was amended and used for each dataset:

https://github.com/Nimitt-nim/NLPAssignment1

Dataset Information

Downloaded Datasets

NGrams Bengali Dataset 2015-17

The dataset has scraped data from around **21,500** internet sources. The dataset is very scattered evident from its extention (.raw) and needed sorting and organising.

Link to dataset: https://data.statmt.org/ngrams/raw/

Link to processed dataset after all steps

https://huggingface.co/datasets/nimittnim/NGrams-Bengali

The following link directs to CSV file documenting all the sources with number of articles in each source : https://github.com/Nimitt-nim/NLPAssignment1/blob/main/sources.csv

The following table shows the amount of data at each step :

Task	Volume of Dataset (GB)
Downloading	22.9
Preprocessing	20.1
Cleaning	19.13
Deduplication 1	15.3
Deduplication 2	8.2

Other Datasets

1. The dataset was downloaded from the following link : https://opus.nlpl.eu/CCMatrix/bn&/v1/CCMatrix#download

   The following table shows the amount of data at each step :

Task	Volume of Dataset (GB)
Downloading	11.2
Preprocessing	11
Cleaning	8.6
Deduplication 1	6.31

2. The dataset was downloaded from the following link :

   https://zenodo.org/records/11111869

   This dataset has over 1.9 million news articles scraped from the internet.

   The following table shows the amount of data at each step :

Task	Volume of Dataset (GB)
Downloading	16.7
Preprocessing	11.9
Cleaning	11.6
Deduplication 1	10.7

3. The dataset was downloaded from the following link :

   https://www.kaggle.com/datasets/furcifer/bangla-newspaper-dataset?resource=download

   This dataset has over 400k+ news articles scraped from the internet.

   The following table shows the amount of data at each step :

Task	Volume of Dataset (GB)
Downloading	6.7
Preprocessing	5.9
Cleaning	3.76
Deduplication 1	1.98

The final processed files can be found on the following link :
https://huggingface.co/datasets/HarshitIITGN/NLP_bn

Scraped Data

The dataset contains text data scraped from 16 major Bengali newspaper websites and 10,000 links from Bengali Wikipedia. The text files containing the scraped data can be found at the following link:
https://huggingface.co/datasets/HarshitIITGN/NLP_bn

The following table shows the amount of data at each step :

Task	Volume of Dataset (GB)
Scraping	5.7
Preprocessing	5.65

Cleaning	5.59
Deduplication 1	5.56

Sorting and Organising

The datasets were sorted according to sources. A directory was formed for each source containing all the articles and pages data into text files. The basic layout of the program to achieve this is similar to the below code. But was adapted according to each dataset.

```
import re
import os

# Directory to store the output files
output_dir = 'NGramsSortedData'
os.makedirs(output_dir, exist_ok=True)  # Create the directory i


raw_data_dir = 'NGramsData'
raw_files = [f for f in os.listdir(raw_data_dir) if f.endswith(
for raw_name in raw_files:
    # Open the input file in read mode
    with open(f"{raw_data_dir}/{raw_name}", 'r', encoding='utf-8
        uri = 'https://0init/'  # Placeholder URI to start
        lines_to_write = []

        # Read the input file line by line
        for line in infile:
            if "uri:" in line:
                # If we encounter a new URI, write the collected
                if uri and lines_to_write:
                    # Extract the domain from the URI
                    domain_match = re.search(r'//([^/]+)/', uri)
                    if domain_match:
```

```
                    domain = domain_match.group(1)  # Extrac

                    # Replace invalid characters in the doma
                    filename = re.sub(r'[^\w]', '_', domain)

                    # Write the data to a domain-specific fi
                    with open(os.path.join(output_dir, filen
                        outfile.write("\n-------------------
                        outfile.writelines(lines_to_write)
                        outfile.write("-------------------

            # Extract the new URI from the current line
            uri_match = re.search(r'uri:(http[^\s]+)', line)
            if uri_match:
                uri = uri_match.group(1)  # Extract the full

            # Reset the buffer for the next URI
            lines_to_write = []

        # Collect other data related to the URI
        else:
            lines_to_write.append(line)

    # Write the last URI and its corresponding data (if any)
    if uri and lines_to_write:
        domain_match = re.search(r'//([^/]+)/', uri)
        if domain_match:
            domain = domain_match.group(1)
            filename = re.sub(r'[^\w]', '_', domain) + '.txt

            with open(os.path.join(output_dir, filename), 'a
                outfile.write("\n-----------------------\n")
                outfile.writelines(lines_to_write)
                outfile.write("-----------------------\n")
```

```
    print(f"Processing complete for raw{raw_name}")
```

Preprocessing

In preprocessing, all the irrelevant spaces and english alphabets were removed. The following root code was adopted based on dataset processed.

```
import os
import re
import pandas as pd

# Global variables for words
porn_words = []
bad_bengali_words = []

# Function to load words from a CSV file into a global list
def load_words_from_csv(file_path, word_list):
    df = pd.read_csv(file_path)
    word_list.extend(df.iloc[:, 0].tolist())  # Assuming words a

# Function to check if the filename has any pornographic Englis
def check_filename_for_porn(filename):
    for word in porn_words:
        if word.lower() in filename.lower():
            return True
    return False

# Function to remove English characters from text
def remove_english_characters(text):
    return re.sub(r'[a-zA-Z]', '', text)

# Function to remove bad Bengali words
def remove_bad_bengali_words(text):
    for word in bad_bengali_words:
```

```
        text = text.replace(word, '')
    return text

# Main function to process a single file
def process_file(input_filename, output_filename):
    # Check if filename contains any pornographic words
    if check_filename_for_porn(input_filename):
        print(f"Filename '{input_filename}' contains inappropria
        return

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

    # Remove English characters
    content = remove_english_characters(content)

    # Remove bad Bengali words
    content = remove_bad_bengali_words(content)

    # Write the cleaned content to a new file
    with open(output_filename, 'w', encoding='utf-8') as output_
        output_file.write(content)

# Function to load words and process files in a folder
def process_files_in_folder(input_folder, output_folder, porn_wo
    # Load the lists of words
    load_words_from_csv(porn_words_file, porn_words)
    load_words_from_csv(bad_bengali_words_file, bad_bengali_word

    # Create output folder if it doesn't exist
    os.makedirs(output_folder, exist_ok=True)

    # Process each file in the input folder
    for filename in os.listdir(input_folder):
        if filename.endswith(".txt"):  # Process only text files
```

```
            input_file_path = os.path.join(input_folder, filenar
            output_file_path = os.path.join(output_folder, filer
            process_file(input_file_path, output_file_path)

# Example usage
input_folder = "NGramsSortedData"  # Replace with your input fol
output_folder = "NGramsCleanedData"  # Output folder name
porn_words_file = "porn_words_english.csv"  # Replace with your
bad_bengali_words_file = "bad_words_bengali.csv"  # Replace witl

process_files_in_folder(input_folder, output_folder, porn_words_
```

Pornographic Cleaning

Articles were removed that contained any pornographic content.

Step 1: First, the domain name (of available) of the source of data was checked for any pornographic words in English.

Step 2: Bad words were removed using regular expressions

```
import os
import re
import pandas as pd

# Global variables for words
porn_words = []
bad_bengali_words = []

# Function to load words from a CSV file into a global list
def load_words_from_csv(file_path, word_list):
    df = pd.read_csv(file_path)
    word_list.extend(df.iloc[:, 0].tolist())  # Assuming words a

# Function to check if the filename has any pornographic Englisl
def check_filename_for_porn(filename):
```

```
    for word in porn_words:
        if word.lower() in filename.lower():
            return True
    return False

# Function to remove English characters from text
def remove_english_characters(text):
    return re.sub(r'[a-zA-Z]', '', text)

# Function to remove bad Bengali words
def remove_bad_bengali_words(text):
    for word in bad_bengali_words:
        text = text.replace(word, '')
    return text

# Main function to process a single file
def process_file(input_filename, output_filename):
    # Check if filename contains any pornographic words
    if check_filename_for_porn(input_filename):
        print(f"Filename '{input_filename}' contains inappropria
        return

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

    # Remove English characters
    content = remove_english_characters(content)

    # Remove bad Bengali words
    content = remove_bad_bengali_words(content)

    # Write the cleaned content to a new file
    with open(output_filename, 'w', encoding='utf-8') as output_
        output_file.write(content)
```

```
# Function to load words and process files in a folder
def process_files_in_folder(input_folder, output_folder, porn_wo
    # Load the lists of words
    load_words_from_csv(porn_words_file, porn_words)
    load_words_from_csv(bad_bengali_words_file, bad_bengali_word

    # Create output folder if it doesn't exist
    os.makedirs(output_folder, exist_ok=True)

    # Process each file in the input folder
    for filename in os.listdir(input_folder):
        if filename.endswith(".txt"):  # Process only text files
            input_file_path = os.path.join(input_folder, filenam
            output_file_path = os.path.join(output_folder, filen
            process_file(input_file_path, output_file_path)

# Example usage
input_folder = "NGramsSortedData"  # Replace with your input fol
output_folder = "NGramsCleanedData"  # Output folder name
porn_words_file = "porn_words_english.csv"  # Replace with your
bad_bengali_words_file = "bad_words_bengali.csv"  # Replace with

process_files_in_folder(input_folder, output_folder, porn_words_
```

Deduplication

Method 1 : Removing exact duplicate articles/webpages by SHA hashing

Hashing each article and removing duplicate articles if it corresponds to already hashed index.

```
import hashlib
import os
```

```
# Function to generate a hash for a given text
def generate_hash(text):
    return hashlib.sha256(text.encode('utf-8')).hexdigest()

# Deduplicate articles in multiple text files and save each file
def deduplicate_across_files(input_folder, output_folder, separa
    seen_hashes = set()

    # Loop through all files in the input folder
    for filename in os.listdir(input_folder):
        file_path = os.path.join(input_folder, filename)

        # Ensure it's a text file (optional check)
        if os.path.isfile(file_path) and filename.endswith('.txt
            unique_articles = []
            # Read the file content
            with open(file_path, 'r', encoding='utf-8') as file
                content = file.read()

            # Split content into articles using the separator
            articles = content.split(separator)
            print(articles)
            # Deduplicate the articles
            for article in articles:
                article = article.strip()  # Clean up whitespace
                if article:  # Ensure article is not empty
                    text_hash = generate_hash(article)

                    # If the hash is not in seen_hashes, add it
                    if text_hash not in seen_hashes:
                        seen_hashes.add(text_hash)
                        unique_articles.append(article)

            # Define output file path (same name as input but in
            output_path = os.path.join(output_folder, filename)
```

```
            # Write the unique articles back to the output file
            with open(output_path, 'w', encoding='utf-8') as out
                output_file.write(f"\n{separator}\n".join(unique
                output_file.write(f"\n{separator}\n")  # End wit

# Specify the input folder and output folder
input_folder = 'NGramsCleanedData'  # Replace with your input fo
output_folder = 'NGramsDeduplicatedData'  # Replace with your ou

# Ensure the output folder exists, if not, create it
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

# Call the function
deduplicate_across_files(input_folder, output_folder)

print("Deduplication across files complete!")
```

Method 2: Removing similar matches using SimHash

Removing articles that are very similar to previously seen articles

```
import os
from simhash import Simhash

# Function to deduplicate articles in multiple text files using
def deduplicate_using_simhash(input_folder, output_folder, separ
    seen_hashes = set()
    article_count = 0
    unique_count = 0

    # Loop through all files in the input folder
    for filename in os.listdir(input_folder):
```

```
        file_path = os.path.join(input_folder, filename)

        # Ensure it's a text file (optional check)
        if os.path.isfile(file_path) and filename.endswith('.txt
            unique_articles = []
            # Read the file content
            with open(file_path, 'r', encoding='utf-8') as file
                content = file.read()

            # Split content into articles using the separator
            articles = content.split(separator)
            articles = [article.strip() for article in articles

            # Deduplicate the articles
            for article in articles:
                article_hash = Simhash(article)
                article_count += 1
                is_duplicate = False

                # Check against already seen hashes
                for seen_hash in seen_hashes:
                    if article_hash.distance(Simhash(seen_hash))
                        is_duplicate = True
                        break

                if not is_duplicate:
                    seen_hashes.add(article_hash.value)  # Store
                    unique_articles.append(article)
                    unique_count += 1

            # Define output file path (same name as input but in
            output_path = os.path.join(output_folder, filename)

            # Write the unique articles back to the output file
            with open(output_path, 'w', encoding='utf-8') as out
                output_file.write(f"\n{separator}\n".join(unique
```

```
                output_file.write(f"\n{separator}\n")  # End wi

    print(f"Processed {article_count} articles. Found {unique_c

# Specify the input folder and output folder
input_folder = 'NGramsDeduplicated1'  # Replace with your input
output_folder = 'NGramsDeduplicated2'  # Replace with your outpu

# Ensure the output folder exists, if not, create it
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

# Call the function
deduplicate_using_simhash(input_folder, output_folder)

print("Deduplication using Simhash complete!")
```

Method 3: Removing similar matches using LSH

```
import os
from datasketch import MinHash, MinHashLSH
import hashlib

def minhash_signature(article, num_perm=128):
    """Generate a MinHash signature for an article."""
    m = MinHash(num_perm=num_perm)
    for word in article.split():
        m.update(word.encode('utf8'))
    return m

def deduplicate_using_minhash(input_folder, output_folder, separ
    unique_articles_by_file = {}
```

```
    filenames = [f for f in os.listdir(input_folder) if f.endswi

    for filename in filenames:
        file_path = os.path.join(input_folder, filename)

        if os.path.isfile(file_path):
            with open(file_path, 'r', encoding='utf-8') as file
                content = file.read()

            articles = content.split(separator)
            articles = [article.strip() for article in articles

            if articles:
                lsh = MinHashLSH(threshold=similarity_threshold,
                seen_articles = {}

                for i, article in enumerate(articles):
                    sig = minhash_signature(article)
                    # Check for similar articles
                    similar = lsh.query(sig)

                    if not similar:  # No similar articles found
                        lsh.insert(f'article_{i}', sig)  # Inser
                        unique_articles_by_file.setdefault(filen
                        seen_articles[f'article_{i}'] = article


            # Ensure unique articles are saved back
            if filename not in unique_articles_by_file:
                unique_articles_by_file[filename] = []

            output_path = os.path.join(output_folder, filename)
            with open(output_path, 'w', encoding='utf-8') as out
                output_file.write(f"\n{separator}\n".join(unique
                output_file.write(f"\n{separator}\n")
```

```
# Specify folders
input_folder = 'NGramsDeduplicated1'  # Replace with your input
output_folder = 'NGramsDeduplicated2'  # Replace with your outpu


# Ensure output folder exists
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

# Call the deduplication function
deduplicate_using_minhash(input_folder, output_folder)

print("MinHash Deduplication complete!")
```

Data Storage

All the datasets were stored on hugging face as compressed files.

Acknowledgement

We would like to thank Prof. Mayank Singh for providing the opportunity to work on the assignment. We also thanks Teaching Assistants for the course for conducting hands on sessions and assisting throughout the assignment.

Work Distribution

Member Name : Nimitt

- Downloaded and organised NGrams Bengali Data
- Wrote code for Bad words and articles removal for NGrams data
- Wrote code for Preprocessing
- Implemented Deduplication of NGrams data
- (Earlier Being part of Team 7) Scraped 1GB of Assamese data from various news websites, Internet Archive and Wikipedia. Downloaded 20GB Assamese

data from Ai4Bharat and organised it.

Member Name : Harshit

- Scraped 6GB of data from various major bengali newspaper’s websites and wikipedia links.
- Wrote code for bad words and article removal for the scraped, opus, kaggle and zenedo datasets.
- Wrote code for Preprocessing.
- Implemented deduplication of the above mentioned datasets.

Member Name: Pratham Sharda

- Scraped 20GB of data from various sources like News websites and Internet Archive

Member Name: Pranjal Gaur

- Downloaded and processed NGrams Data

Member Name: Shubham Aggarwal

- Wrote code for Bad words removal

Member Name: Chirag Patel

- Downloaded Oscars dataset