AIWIR Lab Week 3

Team-8

Team members :

1. A Spoorthi Alva PES2UG19CS001
2. A R Manyatha PES2UG19CS002
3. Achyut Jagini. PES2UG19CS013
4. Amulya S Dinesh PES2UG19CS035

Dataset: https://www.kaggle.com/moupriyaroy/20-newsgroups

Tasks :

- ➔ Removing stop words
- ➔ Removing punctuation
- ➔ Convert to lowercase
- ➔ Stemming
- ➔ Converting number its equivalent words
- ➔ Removing header

Tool used : Pycharm

- ➔ It is installed on the computer.
- ➔ The dataset is downloaded and added to the folder where the main program exists
- ➔ Project is created in Pycharm by opening a folder containing the dataset.
- ➔ Required modules are installed.

Libraries used :

- ➔ NLTK
- ➔ Pandas
- ➔ NumPy
- ➔ Pickle

Steps to build a Positional Index

- ➔ Fetch the document.
- ➔ Remove stop words, stem the resulting words.
- ➔ If the word is already present in the dictionary, add the document and the corresponding positions it appears in. Else, create a new entry.

- ➔ Also update the frequency of the word for each document, as well as the no. of documents it appears in.

Code :

```
# importing libraries
import numpy as np
import os
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import TweetTokenizer
from natsort import natsorted
import string

def read_file(filename):
    with open(filename, 'r', encoding="ascii", errors="surrogateescape") as f:
        stuff = f.read()
    f.close()

    # Remove header and footer.
    stuff = remove_header_footer(stuff)
    return stuff

def remove_header_footer(final_string):
    new_final_string = ""
```

```
 tokens = final_string.split('\n\n')

    # Remove tokens[0] and tokens[-1]
    for token in tokens[1:-1]:
        new_final_string += token + " "
    return new_final_string

def preprocessing(final_string):
    # Tokenize.
    tokenizer = TweetTokenizer()
    token_list = tokenizer.tokenize(final_string)
    # Remove punctuations.
    table = str.maketrans('', '', '\t')
    token_list = [word.translate(table) for word in token_list]
    punctuations = (string.punctuation).replace("'", "")
    trans_table = str.maketrans('', '', punctuations)
    stripped_words = [word.translate(trans_table) for word in token_list]
    token_list = [str for str in stripped_words if str]

    # Change to lowercase.
    token_list = [word.lower() for word in token_list]
    return token_list

# In this example, we create the positional index for only 1 folder.
folder_names = ["comp.graphics"]
```

```
# Initialize the stemmer.
stemmer = PorterStemmer()

# Initialize the file no.
fileno = 0
pos_index = {}

# Initialize the file mapping (fileno -> file name).
file_map = {}

for folder_name in folder_names:
    file_names = natsorted(os.listdir("20_newsgroups/" + folder_name))
    for file_name in file_names:
        stuff = read_file("20_newsgroups/" + folder_name + "/" + file_name)
      final_token_list = preprocessing(stuff)
        for pos, term in enumerate(final_token_list):
            term = stemmer.stem(term)

            # If the term already exists in the positional index dictionary.
            if term in pos_index:
                n='0123456789'
                pos_index[term][0] = pos_index[term][0] + 1

                # Check if the term has existed in that DocID before.
```

```
            if fileno in pos_index[term][1]:
                pos_index[term][1][fileno].append(pos)

            else:
                pos_index[term][1][fileno] = [pos]

        # If term does not exist in the positional index dictionary (first encounter).
        else:
            pos_index[term] = [].
            pos_index[term].append(1)
            pos_index[term].append({})
            # Add doc ID to postings list.
            pos_index[term][1][fileno] = [pos]

    # Map the file no. to the file name.
    file_map[fileno] = "20_newsgroups/" + folder_name + "/" + file_name

    # Increment the file no. counter for document ID mapping
    fileno += 1

# Sample positional index to test the code.
sample_pos_idx = pos_index["andrew"]
print("Positional Index")
print(sample_pos_idx)
```

```
file_list = sample_pos_idx[1]
print("Filename, [Positions]")
for fileno, positions in file_list.items():
    print(file_map[fileno], positions)


#Converting numbers to words
def convert_to_words(num):
        # Get number of digits  in given number
        l = len(num)
        # Base cases
        if (l == 0):
                print("empty string")
                return
        if (l > 4):
                print("Length more than 4 is not supported")
                return
        single_digits = ["zero", "one", "two", "three","four", "five", "six", "seven","eight",
"nine"]
        two_digits = ["", "ten", "eleven", "twelve","thirteen", "fourteen",
"fifteen","sixteen", "seventeen", "eighteen","nineteen"]
        tens_multiple = ["", "", "twenty", "thirty", "forty","fifty", "sixty", "seventy",
"eighty","ninety"]
        tens_power = ["hundred", "thousand"]
        print(num, ":", end=" ")
        if (l == 1):
                print(single_digits[ord(num[0]) - 48])
```

```
        return

x = 0
while (x < len(num)):
        if (l >= 3):
                if (ord(num[x]) - 48 != 0):
                        print(single_digits[ord(num[x]) - 48],
                                end=" ")
                        print(tens_power[l - 3], end=" ")
                l -= 1
        else:
                if (ord(num[x]) - 48 == 1):
                        sum = (ord(num[x]) - 48 +
                                ord(num[x+1]) - 48)
                        print(two_digits[sum])
                        return
                elif (ord(num[x]) - 48 == 2 and
                        ord(num[x + 1]) - 48 == 0):
                        print("twenty")
                        return
                else:
                        i = ord(num[x]) - 48
                        if(i > 0):
                                print(tens_multiple[i], end=" ")
                        else:
```

```
                    print("", end="")
                x += 1
                if(ord(num[x]) - 48 != 0):
                    print(single_digits[ord(num[x]) - 48])
        x += 1


h = open('eg.txt', 'r')
content = h.readlines()
for line in content:
    for i in line.split(" "):
        if i.isdigit() == True:
            convert_to_words(i)
```

```python
two_digits = ["", "ten", "eleven", "twelve",
              "thirteen", "fourteen", "fifteen",
              "sixteen", "seventeen", "eighteen",
              "nineteen"]


tens_multiple = ["", "", "twenty", "thirty", "forty",
                 "fifty", "sixty", "seventy", "eighty",
                 "ninety"]

tens_power = ["hundred", "thousand"]

print(num, ":", end=" ")


if (l == 1):
    print(single_digits[ord(num[0]) - 48])
    return

x = 0
while (x < len(num)):

    if (l >= 3):
```

```python
x = 0
while (x < len(num)):

    if (l >= 3):
        if (ord(num[x]) - 48 != 0):
            print(single_digits[ord(num[x]) - 48],
                end=" ")
            print(tens_power[l - 3], end=" ")

        l -= 1

    else:

        if (ord(num[x]) - 48 == 1):
            sum = (ord(num[x]) - 48 +
                ord(num[x+1]) - 48)
            print(two_digits[sum])
            return

        elif (ord(num[x]) - 48 == 2 and
            ord(num[x + 1]) - 48 == 0):
            print("twenty")
            return

        else:
            i = ord(num[x]) - 48
            if(i > 0):
                print(tens_multiple[i], end=" ")
```

```python
                return

            elif (ord(num[x]) - 48 == 2 and
                ord(num[x + 1]) - 48 == 0):
                print("twenty")
                return

            else:
                i = ord(num[x]) - 48
                if(i > 0):
                    print(tens_multiple[i], end=" ")
                else:
                    print("", end="")
                x += 1
                if(ord(num[x]) - 48 != 0):
                    print(single_digits[ord(num[x]) - 48])
        x += 1

h = open('eg.txt', 'r')

content = h.readlines()

for line in content:

    for i in line.split(" "):
        if i.isdigit() == True:
            convert_to_words(i)
```

Output:

```
C:\Users\manya\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/manya/PycharmProjects/pythonProject/main.py
32 : thirty two
45 : forty five
1034 : one thousand thirty four
431 : four hundred thirty one
100 : one hundred
Process finished with exit code 0
```

```python
# importing libraries
import numpy as np
import os
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import TweetTokenizer
from natsort import natsorted
import string

def read_file(filename):
    with open(filename, 'r', encoding="ascii", errors="surrogateescape") as f:
        stuff = f.read()

    f.close()
    # Remove header and footer.
    stuff = remove_header_footer(stuff)

    return stuff

def remove_header_footer(final_string):
    new_final_string = ""
    tokens = final_string.split('\n\n')

    # Remove tokens[0] and tokens[-1]
    for token in tokens[1:-1]:
        new_final_string += token + " "
    return new_final_string

def preprocessing(final_string):
```

```python
def preprocessing(final_string):
    # Tokenize.
    tokenizer = TweetTokenizer()
    token_list = tokenizer.tokenize(final_string)
    # Remove punctuations.
    table = str.maketrans('', '', '\t')
    token_list = [word.translate(table) for word in token_list]
    punctuations = (string.punctuation).replace("'", "")
    trans_table = str.maketrans('', '', punctuations)
    stripped_words = [word.translate(trans_table) for word in token_list]
    token_list = [str for str in stripped_words if str]

    # Change to lowercase.
    token_list = [word.lower() for word in token_list]
    return token_list

# In this example, we create the positional index for only 1 folder.
folder_names = ["comp.graphics"]

# Initialize the stemmer.
stemmer = PorterStemmer()

# Initialize the file no.
fileno = 0

# Initialize the dictionary.
pos_index = {}
```

```python
pos_index = {}

# Initialize the file mapping (fileno -> file name).
file_map = {}

for folder_name in folder_names:

    # Open files.
    file_names = natsorted(os.listdir("20_newsgroups/" + folder_name))

    # For every file.
    for file_name in file_names:

        # Read file contents.
        stuff = read_file("20_newsgroups/" + folder_name + "/" + file_name)

        final_token_list = preprocessing(stuff)

        # For position and term in the tokens.
        for pos, term in enumerate(final_token_list):

            # First stem the term.
            term = stemmer.stem(term)

            # If term already exists in the positional index dictionary.
            if term in pos_index:
                n='0123456789'
                # Increment total freq by 1.
                pos_index[term][0] = pos_index[term][0] + 1
```

```python
            # If term already exists in the positional index dictionary.
            if term in pos_index:
                n='0123456789'
                # Increment total freq by 1.
                pos_index[term][0] = pos_index[term][0] + 1

                # Check if the term has existed in that DocID before.
                if fileno in pos_index[term][1]:
                    pos_index[term][1][fileno].append(pos)

                else:
                    pos_index[term][1][fileno] = [pos]

            # If term does not exist in the positional index dictionary
            # (first encounter).
            else:

                # Initialize the list.
                pos_index[term] = []
                # The total frequency is 1.
                pos_index[term].append(1)
                # The postings list is initially empty.
                pos_index[term].append({})
                # Add doc ID to postings list.
                pos_index[term][1][fileno] = [pos]

        # Map the file no. to the file name.
        file_map[fileno] = "20_newsgroups/" + folder_name + "/" + file_name
```

```python
            # If term does not exist in the positional index dictionary
            # (first encounter).
            else:

                # Initialize the list.
                pos_index[term] = []
                # The total frequency is 1.
                pos_index[term].append(1)
                # The postings list is initially empty.
                pos_index[term].append({})
                # Add doc ID to postings list.
                pos_index[term][1][fileno] = [pos]

        # Map the file no. to the file name.
        file_map[fileno] = "20_newsgroups/" + folder_name + "/" + file_name

        # Increment the file no. counter for document ID mapping
        fileno += 1

# Sample positional index to test the code.
sample_pos_idx = pos_index["andrew"]
print("Positional Index")
print(sample_pos_idx)

file_list = sample_pos_idx[1]
print("Filename, [Positions]")
for fileno, positions in file_list.items():
    print(file_map[fileno], positions)
```

Output :

```
C:\Users\manya\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/manya/PycharmProjects/pythonProject/main.py
Positional Index
[10, {215: [2081], 539: [66], 591: [879], 616: [462, 473], 680: [135], 691: [2081], 714: [4], 809: [333], 979: [0]}]
Filename, [Positions]
20_newsgroups/comp.graphics/38376 [2081]
20_newsgroups/comp.graphics/38701 [66]
20_newsgroups/comp.graphics/38753 [879]
20_newsgroups/comp.graphics/38778 [462, 473]
20_newsgroups/comp.graphics/38842 [135]
20_newsgroups/comp.graphics/38853 [2081]
20_newsgroups/comp.graphics/38876 [4]
20_newsgroups/comp.graphics/38971 [333]
20_newsgroups/comp.graphics/39663 [0]

Process finished with exit code 0
```