

## **Lab 4: Feature Generation**

**Author: Xu Zhao**

**Module: INT104**

**Teacher: Xiaobo Jin**

**Date: 28/4/2021**

# Introduction

In the given dataset, it is required to generate the features of the txt in the files of the dataset.

## Design & Implementation

### Dynamic Programming

#### Design principles

##### 1. Preprocessing

First, use the `os.listdir()` to read the files in the dataset. Next, use the `dict()` to store all the words in the dataset split by space like this:

```
'could', 'be', 'shown', 'that', 'she', 'had', 'herself', 'been', 'guilty', 'of', 'it', '>', 'on', 'do', 'you',  
'regard', 'sworn', 'statements', 'given', 'to', 'a', '>', 'U', 'S', 'Senate', 'committee', 'as', 'equivalent', 'to',  
'toilet', 'paper', '\n', 'Ahemmm', 'It', 'depends', '(', 'For', 'instance', 'if', 'it', 'were', 'the', 'sworn',  
'statements', 'at', 'the', 'Warren', 'Commission', '\n', 'then', 'yes', 'I', 'would', 'say', 'that', 'the',  
'statements', 'were', 'no', 'better', 'than', 'toilet', '\n', 'paper', 'used', 'at', 'that', 'but', 'in', 'most',  
'cases', 'the', 'answer', 'would', 'be', 'no', '\n', 'Will', 'Steeves', 'go', 'at', 'zoo', 'guild', 'org', 'Neil',  
'Hull', 'is', 'GOID', '\n', 'ZOOID', 'BBS', 'Toronto', 'Ontario', 'The', 'Zoo', 'Of', 'Ids', 'GOIDS', 'Rule', '416',  
'132', '7876', 'Solus', 'Patriarchy', 'Tostelli', 'Patriarchy', 'Paten', 'Hague', '18', 'A', 'HSET', '416'
```

Next, read the `stopwords.txt` and store these words in a set. Store the all words in the above dictionary in a new list. Use the code provided in the tasksheet to transform the words in the list into their lower case form and use an if to delete all non-alphabet characters and delete words which appears in the `stopwords.txt`. Finally, use the code provided in the tasksheet to perform word stemming to remove the word suffix. And the finally preprocessing result is like this:

```
'sur', 'punt', 'jurisdict', 'herod', 'happen', 'town', 'herod', 'appreci', 'courtesi', 'wash', 'morri', 'sharp', 'tongu', 'fool', 'suddenli', 'didn', 'funni', 'peopl', 'lose',  
'pressur', 'attir', 'proper', 'royal', 'ass', 'cover', 'herod', 'problem', 'caesar', 'won', 'fool', 'king', 'long', 'caesar', 'let', 'shout', 'll', 'crowd', 'reaction',  
'puzzl', 'want', 'dead', 'didn', 'king', 'jaw', 'want', 'barabba', 'josephu', 'record', 'thought', 'll', 'barabba', 'time', 'king', 'caesar', 'kill', 'madman', 'hell', 'll',  
'kill', 'day', 'pilat', 'fun', 'joke', 'short', 'point', 'trilingu', 'caught', 'gear', 'govern', 'scientif', 'explan', 'doubt', 'superstiti', 'rumor', 'persist', 'day', 'didn',  
'tomb', 'roman', 'squadron', 'guard', 'sophomor', 'doesn', 'doesn', 'recogn', 'kindr', 'spirit', 'truth', 'admit', 'guess', 'haven', 'learn', 'year', 'fred', 'gilham',  
'gilham', 'csl', 'sri', 'peac', 'war', 'hell', 'war', 'hell', 'sens', 'walker', 'perci', 'come', 'rexlex', 'fnal', 'fnal', 'gov', 'subject', 'assur', 'hell', 'organ', 'fnal',  
'ad', 'net', 'line', 'dream', 'great', 'judgment', 'morn', 'dawn', 'trumpet', 'blown', 'dream', 'sinner', 'gather', 'judgment', 'white', 'throne', 'weep', 'wall', 'lost',  
'fold', 'fate', 'fcl', 'cock', 'mountain', 'coney', 'cneven', 'late', 'seal', 'salvat', 'tonight', 'll', 'cave', 'time', 'calinion', 'ola', 'time', 'die', 'great', 'white'
```

##### 2. TFIDF Representation

To calculate  $f_{ik}$ , define a function. Read the document  $i$  from the dataset and process it in the same way above. This part of code is almost the same with the above. Use a for loop to get all the preprocessed words and use an if to check if it is the same with the given word  $k$  and calculate  $f_{ik}$ .

$N$ : the number of documents in the dataset. Use `len()` to calculate, which is 2726.

$N_k$ , the number of documents which contains word  $k$ . Also use a def to create a function to calculate.

To get  $a_{ik}$ , use some math function as well as  $f_{ik}$  and  $n_k$ .

$D$ , the number of unique words. Use a for loop to get all the words from the preprocessing data and put them into a new set. Because `set()` does not allow same words, we can use `len()` to calculate the length of the set, which is  $D$ .  $D$  is 23433

Finally, use  $a_{ik}$  and some other math function to calculate  $A_{ik}$ .

Finally, put the document name into a list. The length is  $N$  2726. The unique word, the length is  $D$  23433. Create a  $N \times D$  matrix and fill in values of  $A_{ik}$ . Use the code provided to save it into a npz document.

## Implementation

```
import os
import math
import numpy as np
from nltk.stem.porter import *

# Read the folder
datasetname = {i: os.listdir("./dataset/" + i) for i in
os.listdir("./dataset")}
#print(datasetname)

# Store all the words in a dictionary
data = dict()
for dirs in datasetname:
    for i in datasetname.get(dirs):
        if os.path.isfile(os.path.join("./dataset/", dirs, i)):
            with open(os.path.join("./dataset/", dirs, i), 'r',
encoding='Latin1') as fp:
                data[os.path.join("./dataset/", dirs, i)] =
re.split(r'(\W)+', fp.read())
#print(data)

# Read stopwords.txt and store in a set
stopwords = set()
with open("./stopwords.txt", 'r', encoding='utf-8') as gp:
    stopwords = set(gp.read().split())

value = list()
for i in data.keys():
    value.append(data[i])
newdata = list()
for i in value:
    for w in i:
        # Delete all non-alphabet characters and transform characters
into lower case
        wd = re.sub(r'^[a-z]', '', w.lower()).strip()
        # Remove space and stopwords
        if wd != '' and wd not in stopwords:
            newdata.append(wd)

# Perform word stemming to remove the word suffix
stemmer = PorterStemmer()
plurals = newdata
singles = [stemmer.stem(plural) for plural in plurals]
```

```

#print(singles)
#print(len(singles))

# Define function fik
def fik(i, k):
    document = dict()
    #for dirs in datasetname:
        #for i in datasetname.get(dirs):
    with open(os.path.join(i), 'r', encoding="Latin1") as fp1:
        document[os.path.join(i)] = re.split(r'(\W)+', fp1.read())
        value1 = list()
        for j in document.keys():
            value1.append(document[j])
        ndata = list()
        for j in value1:
            for x in j:
                xd = re.sub(r'^a-z]', '', x.lower()).strip()
                if xd != '' and xd not in stopwords:
                    ndata.append(xd)
        stemmer1 = PorterStemmer()
        plurals1 = ndata
        singles1 = [stemmer1.stem(plural) for plural in plurals1]
        count = 0
        for x in singles1:
            if x == k:
                count = count + 1
        return count

# Calculate the number of documents
N = len(data)
#print(N)

# Define function nk
def nk(k):
    count = 0
    for i in data.keys():
        if fik(i, k) != 0:
            count = count + 1
    return count

# Define function aik
def aik(i, k):
    return fik(i, k)*math.log(N/nk(k), 10)

```

```

# Calculate the number of unique words
unique = set()
for x in singles:
    unique.add(x)
D = len(unique)
#print(D)

# Define function Aik
def Aik(i, k):
    sum = 0
    for x in unique:
        sum = sum + math.pow(aik(i, x), 2)
    return aik(i, k)/math.pow(sum, 0.5)

# Store the document i in a list
address = list(data.keys())
#print(address)
#print(unique)

# Create the matrix
A = np.ones((N, D))
for i in address:
    for k in unique:
        A[i][k] = Aik(i, k)

# Save the matrix into npz document
np.savez('train-20ng.npz', X=A)

```

## Test

Although it run a long time, the result is below.

