UE19CS332

Algorithms For Web And Information Retrieval

Assignment – 3

NAME : PRIYA MOHATA
SRN : PES2UG19CS301
SECTION: E

Problem Statement :

- Use case folding (convert all upper case characters to lower case characters)
- Perform lemmatization
- Perform stemming . Now construct the positional index.

TRAIN.CSV
CASE FOLDING :

```
train_data=pd.read_csv('/content/drive/MyDrive/nlp-getting-started/train.csv')
# Case Folding :
train_data["text"] = train_data["text"].str.lower()
train_data.head()
```

	id	keyword	location	text	target
0	1	NaN	NaN	our deeds are the reason of this #earthquake m...	1
1	4	NaN	NaN	forest fire near la ronge sask. canada	1
2	5	NaN	NaN	all residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	just got sent this photo from ruby #alaska as ...	1

STEMMING :

```
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer

# Stemming :
final_train_stem_list=[]
ps = PorterStemmer()
for line in train_data['stop_words_removed']:
  Stem_words=[]
  for i in line:
    rootWord = ps.stem(i)
    Stem_words.append(rootWord)
  Stem_words= [word for word in Stem_words if word.isalnum()]
  final_train_stem_list.append(Stem_words)

print(final_train_stem_list[0:5])
```

Python

```
[['deed', 'reason', 'earthquak', 'may', 'allah', 'forgiv', 'us'], ['forest', 'fire', 'near', 'la', 'rong', 'sask', 'canada'], ['resid',
'ask', 'place', 'notifi', 'offic', 'evacu', 'shelter', 'place', 'order', 'expect'], ['peopl', 'receiv', 'wildfir', 'evacu', 'order',
'california'], ['got', 'sent', 'photo', 'rubi', 'alaska', 'smoke', 'wildfir', 'pour', 'school']]
```

```
train_data['stemmed_words']=final_train_stem_list
train_data.head()
```

	id	keyword	location	text	target	sent_token	word_token	stop_words_removed	stemmed_words
0	1	NaN	NaN	our deeds are the reason of this #earthquake m...	1	[our deeds are the reason of this #earthquake ...	[our, deeds, are, the, reason, of, this, #, ea...	[deeds, reason, #, earthquake, may, allah, for...	[deed, reason, earthquak, may, allah, forgiv, us]
1	4	NaN	NaN	forest fire near la ronge sask. canada	1	[forest fire near la ronge sask., canada]	[forest, fire, near, la, ronge, sask, ., canada]	[forest, fire, near, la, ronge, sask, ., canada]	[forest, fire, near, la, rong, sask, canada]
2	5	NaN	NaN	all residents asked to 'shelter in place' are ...	1	[all residents asked to 'shelter in place' are...	[all, residents, asked, to, 'shelter, in, plac...	[residents, asked, 'shelter, place, ', notifie...	[resid, ask, place, notifi, offic, evacu, shel...
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	[13,000 people receive #wildfires evacuation o...	[13,000, people, receive, #, wildfires, evacua...	[13,000, people, receive, #, wildfires, evacua...	[peopl, receiv, wildfir, evacu, order, califor...
4	7	NaN	NaN	just got sent this photo from ruby #alaska as ...	1	[just got sent this photo from ruby #alaska as...	[just, got, sent, this, photo, from, ruby, #, ...	[got, sent, photo, ruby, #, alaska, smoke, #, ...	[got, sent, photo, rubi, alaska, smoke, wildfi...

LEMMATIZATION:

```
#text after applying lemmatization
import nltk
nltk.download('wordnet')
final_train_lemma_word = []
wordnet_lemmatizer = WordNetLemmatizer()
for line in train_data['stop_words_removed']:
  lemma_word=[];
  for w in line:
      word1 = wordnet_lemmatizer.lemmatize(w, pos = "n")
      word2 = wordnet_lemmatizer.lemmatize(word1, pos = "v")
      word3 = wordnet_lemmatizer.lemmatize(word2, pos = ("a"))
      lemma_word.append(word1)
  lemma_word= [word for word in lemma_word if word.isalnum()]
  final_train_lemma_word.append(lemma_word)

print(final_train_lemma_word[0:5])
```

Python

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[['deed', 'reason', 'earthquake', 'may', 'allah', 'forgive', 'u'], ['forest', 'fire', 'near', 'la', 'ronge', 'sask', 'canada'],
['resident', 'asked', 'place', 'notified', 'officer', 'evacuation', 'shelter', 'place', 'order', 'expected'], ['people', 'receive',
'wildfire', 'evacuation', 'order', 'california'], ['got', 'sent', 'photo', 'ruby', 'alaska', 'smoke', 'wildfire', 'pours', 'school']]
```

```
train_data['lemmatized_words']=final_train_lemma_word
train_data.head()
```

Python

	id	keyword	location	text	target	sent_token	word_token	stop_words_removed	stemmed_words	lemmatized_words
0	1	NaN	NaN	our deeds are the reason of this #earthquake m...	1	[our deeds are the reason of this #earthquake ...	[our, deeds, are, the, reason, of, this, #, ea...	[deeds, reason, #, earthquake, may, allah, for...	[deed, reason, earthquak, may, allah, forgi...	[deed, reason, earthquake, may, allah, forgive...
1	4	NaN	NaN	forest fire near la ronge sask. canada	1	[forest fire near la ronge sask., canada]	[forest, fire, near, la, ronge, sask, ., canada]	[forest, fire, near, la, ronge, sask, ., canada]	[forest, fire, near, la, rong, sask, ., canada]	[forest, fire, near, la, ronge, sask, canada]
2	5	NaN	NaN	all residents asked to 'shelter in place' are ...	1	[all residents asked to 'shelter in place' are...	[all, residents, asked, to, 'shelter, in, plac...	[residents, asked, 'shelter, place, ', notifie...	[resid, ask, 'shelter, place, ', notifi, offic...	[resident, asked, place, notified, officer, ev...
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	[13,000 people receive #wildfires evacuation o...	[13,000, people, receive, #, wildfires, evacua...	[13,000, people, receive, #, wildfires, evacua...	[13,000, peopl, receiv, #, wildfir, evacu, ord...	[people, receive, wildfire, evacuation, order,...
4	7	NaN	NaN	just got sent this photo from ruby #alaska as ...	1	[just got sent this photo from ruby #alaska as...	[just, got, sent, this, photo, from, ruby, #, ...	[got, sent, photo, ruby, #, alaska, smoke, #, ...	[got, sent, photo, rubi, #, alaska, smoke, #, ...	[got, sent, photo, ruby, alaska, smoke, wildfi...

+ Code + Markdown

GENERATING INVERTED INDEX:

```python
# Generating inverted index
def generate_inverted_index(data: list):
    inv_idx_dict = {}
    for index, doc_text in enumerate(data):
        for word in doc_text:
            if word not in inv_idx_dict.keys():
                inv_idx_dict[word] = [index]
            elif index not in inv_idx_dict[word]:
                inv_idx_dict[word].append(index)
    return inv_idx_dict

final_train=generate_inverted_index(final_train_stem_list)
print(final_train)
```

Python

```
{'deed': [0, 4985], 'reason': [0, 304, 305, 317, 319, 746, 763, 781, 894, 1920, 2112, 2252, 2747, 3452, 4084, 4333, 4669, 4843, 4991,
4997, 5000, 5001, 5372, 6166, 6232, 6242, 6453, 6459, 6898, 6991, 7218], 'earthquak': [0, 3027, 3028, 3029, 3030, 3032, 3033, 3037,
3038, 3039, 3041, 3043, 3044, 3046, 3047, 3048, 3049, 3050, 3051, 3054, 3058, 3059, 3060, 3061, 3062, 3064, 3065, 5737, 6024, 6030,
6945, 6946, 6954, 6972, 6974, 7129, 7131, 7132, 7135, 7136, 7142, 7143, 7310, 7589, 7599], 'may': [0, 240, 807, 894, 938, 1475, 1633,
1832, 1934, 1935, 1976, 2001, 2014, 2015, 2114, 2341, 2374, 2399, 2462, 2784, 2793, 2996, 3025, 3112, 3221, 3393, 3399, 3723, 3773,
4010, 4138, 4140, 4177, 4215, 4232, 4235, 4255, 4516, 4562, 4719, 4723, 4816, 4954, 5010, 5053, 5156, 5244, 5245, 5247, 5248, 5249,
5250, 5251, 5253, 5257, 5259, 5260, 5262, 5264, 5266, 5269, 5270, 5271, 5272, 5273, 5274, 5275, 5276, 5277, 5367, 5394, 5410, 5471,
5661, 5801, 5839, 5852, 6467, 6484, 6500, 6532, 6646, 7003, 7083, 7382, 7601], 'allah': [0, 3725, 4010, 4255, 4290, 4299, 4312, 4679,
6447], 'forgiv': [0, 2265, 2532], 'us': [0, 42, 84, 268, 283, 337, 456, 468, 470, 513, 627, 628, 681, 829, 855, 1035, 1099, 1115, 1157,
```

GENERATING POSITIONAL INDEX:

```python
# Positional Index :
# Initialize the dictionary.
pos_index = {}
# Initialize the file mapping (fileno -> file name).
file_map = {}
def generate_positional_index(data:list):
  fileno=0
  lineno=-1
  for line in data:
    lineno+=1;
    for pos, term in enumerate(line):
      if term in pos_index:
        # 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][lineno].append(pos)
        else:
          pos_index[term][1][lineno] = [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][lineno] = [pos]
      fileno += 1
  return pos_index

final=generate_positional_index(final_train_stem_list)
count=0
for i in final:
  count=count+1;
```

```python
final=generate_positional_index(final_train_stem_list)
count=0
for i in final:
  count=count+1;
  if count<=5:
    print(i,final[i])
  else:
    break;
```

Python

```
deed [2, {0: [0], 4985: [8]}]
reason [31, {0: [1], 304: [6], 305: [6], 317: [5], 319: [6], 746: [4], 763: [2], 781: [6], 894: [5], 1920: [0], 2112: [7], 2252: [1],
2747: [9], 3452: [0], 4084: [4], 4333: [5], 4669: [0], 4843: [8], 4991: [1], 4997: [1], 5000: [1], 5001: [1], 5372: [8], 6166: [0],
6232: [0], 6242: [2], 6453: [8], 6459: [2], 6898: [7], 6991: [0], 7218: [5]}]
earthquak [51, {0: [2], 3027: [0], 3028: [8], 3029: [3], 3030: [8], 3032: [6], 3033: [8], 3037: [7], 3038: [7], 3039: [5], 3041: [3],
3043: [9], 3044: [1], 3046: [2], 3047: [5], 3048: [1], 3049: [7], 3050: [0], 3051: [3], 3054: [0], 3058: [0], 3059: [11], 3060: [0],
3061: [0], 3062: [1], 3064: [0], 3065: [4], 5737: [4], 6024: [3], 6030: [11], 6945: [12], 6946: [13], 6954: [0], 6972: [3], 6974: [12],
7129: [0], 7131: [6], 7132: [7], 7135: [2], 7136: [6], 7142: [11], 7143: [7], 7310: [8], 7589: [1], 7599: [8]}]
may [87, {0: [3], 240: [4], 807: [8], 894: [3], 938: [4], 1475: [7], 1633: [8], 1832: [8], 1934: [1], 1935: [13], 1976: [0], 2001: [8],
2014: [1], 2015: [8], 2114: [6], 2341: [5], 2374: [1], 2399: [0], 2462: [13], 2784: [6], 2793: [2], 2996: [10], 3025: [10], 3112: [5],
3221: [3], 3393: [5], 3399: [6], 3723: [9], 3773: [0], 4010: [0], 4138: [3], 4140: [2], 4177: [3], 4215: [10], 4232: [2], 4235: [2],
4255: [8], 4516: [5], 4562: [1], 4719: [0], 4723: [0], 4816: [2], 4954: [2], 5010: [6], 5053: [3], 5156: [4], 5244: [3], 5245: [5],
5247: [3], 5248: [3], 5249: [3], 5250: [3], 5251: [3], 5253: [4], 5257: [5], 5259: [3], 5260: [3], 5262: [3], 5264: [4], 5266: [4],
5269: [3], 5270: [3], 5271: [4], 5272: [3], 5273: [3], 5274: [4], 5275: [3], 5276: [3], 5277: [5], 5367: [9], 5394: [9], 5410: [1],
5471: [11], 5661: [4], 5801: [5], 5839: [3], 5852: [2], 6467: [3], 6484: [0], 6500: [2], 6532: [3], 6646: [0], 7003: [3], 7083: [8],
7382: [11], 7601: [5]}]
allah [9, {0: [4], 3725: [5], 4010: [1], 4255: [9], 4290: [0], 4299: [0], 4312: [0], 4679: [7], 6447: [6]}]
```

TEST.CSV

CASE FOLDING :

```python
test_data=pd.read_csv('/content/drive/MyDrive/nlp-getting-started/test.csv')
test_data["text"] = test_data["text"].str.lower()
test_data.head()
```

	id	keyword	location	text
0	0	NaN	NaN	just happened a terrible car crash
1	2	NaN	NaN	heard about #earthquake is different cities, s...
2	3	NaN	NaN	there is a forest fire at spot pond, geese are...
3	9	NaN	NaN	apocalypse lighting. #spokane #wildfires
4	11	NaN	NaN	typhoon soudelor kills 28 in china and taiwan

+ Code + Markdown

STEMMING :

```python
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer

# Stemming :
final_test_stem_list=[]
ps = PorterStemmer()
for line in test_data['stop_words_removed']:
  Stem_words=[]
  for i in line:
    rootWord = ps.stem(i)
    Stem_words.append(rootWord)
  Stem_words= [word for word in Stem_words if word.isalnum()]
  final_test_stem_list.append(Stem_words)

print(final_test_stem_list[0:5])
```

Python

```
[['happen', 'terribl', 'car', 'crash'], ['heard', 'earthquak', 'differ', 'citi', 'stay', 'safe', 'everyon'], ['forest', 'fire', 'spot',
'pond', 'gees', 'flee', 'across', 'street', 'save'], ['apocalyps', 'light', 'spokan', 'wildfir'], ['typhoon', 'soudelor', 'kill', '28',
'china', 'taiwan']]
```

LEMMATIZATION :

```python
#text after applying lemmatization
import nltk
final_test_lemma_word = []
wordnet_lemmatizer = WordNetLemmatizer()
for line in test_data['stop_words_removed']:
  lemma_word=[];
  for w in line:
      word1 = wordnet_lemmatizer.lemmatize(w, pos = "n")
      word2 = wordnet_lemmatizer.lemmatize(word1, pos = "v")
      word3 = wordnet_lemmatizer.lemmatize(word2, pos = ("a"))
      lemma_word.append(word1)
  lemma_word= [word for word in lemma_word if word.isalnum()]
  final_test_lemma_word.append(lemma_word)
print(final_test_lemma_word[0:5])
```

Python

```
[['happened', 'terrible', 'car', 'crash'], ['heard', 'earthquake', 'different', 'city', 'stay', 'safe', 'everyone'], ['forest', 'fire',
'spot', 'pond', 'goose', 'fleeing', 'across', 'street', 'save'], ['apocalypse', 'lighting', 'spokane', 'wildfire'], ['typhoon',
'soudelor', 'kill', '28', 'china', 'taiwan']]
```

GENERATING INVERTED INDEX:

```python
final_test=generate_inverted_index(final_test_stem_list)
print(final_test)
```

Python

```
{'happen': [0, 37, 42, 48, 66, 170, 310, 457, 502, 526, 537, 694, 822, 891, 955, 1040, 1110, 1129, 1138, 1203, 1323, 1413, 1438, 1497,
1878, 1911, 2139, 2568, 2638, 2663, 3027, 3039, 3068, 3216, 3229], 'terribl': [0, 131, 510, 1856, 2043, 2675, 2969], 'car': [0, 32, 78,
79, 88, 89, 230, 527, 676, 738, 765, 775, 787, 789, 795, 803, 814, 1062, 1088, 1152, 1240, 1422, 1480, 1486, 1491, 1655, 1657, 1659,
1849, 2366, 2682, 2802, 2928, 2956, 3125, 3208, 3212, 3219, 3243, 3257], 'crash': [0, 75, 84, 775, 776, 777, 778, 779, 780, 782, 783,
784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 1041, 1538,
2045, 2105, 2111, 2116, 2484, 2485, 2486, 2488, 2489, 2491, 2498, 2499, 2501, 2502, 2503, 2638, 2716, 2717, 2718, 2719, 2720, 2723,
2724, 2726, 2728, 2729, 2814], 'heard': [1, 145, 747, 954, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2036, 2107,
2216, 2476, 2755, 2908], 'earthquak': [1, 5, 1100, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1329, 1330, 1548, 2160, 2161, 2585, 3020,
3021, 3030, 3258], 'differ': [1, 750, 1609, 2618, 2730, 2846], 'citi': [1, 27, 471, 491, 530, 537, 562, 734, 877, 927, 1182, 1380,
1382, 1389, 1390, 1391, 1392, 1546, 1622, 2225, 2606, 2773, 3065, 3071, 3259], 'stay': [1, 737, 1036, 1337, 1642, 1740, 1800, 2730,
3007, 3077, 3152, 3210], 'safe': [1, 556, 1378, 1446, 1620, 1740, 1968, 2936, 3158], 'everyon': [1, 317, 497, 562, 629, 910, 1040,
1740, 1857, 2313, 2317, 2538, 2698, 3097, 3250], 'forest': [2, 106, 109, 110, 114, 115, 117, 118, 120, 121, 122, 123, 1140, 1149, 1691,
```

GENERATING POSITION INDEX:

```
happen [35, {0: [0], 37: [11], 42: [1], 48: [1], 66: [2], 170: [4], 310: [0], 457: [10], 502: [0], 526: [2], 537: [2], 694: [7], 822:
[10], 891: [8], 955: [2], 1040: [8], 1110: [2], 1129: [7], 1138: [8], 1203: [4], 1323: [2], 1413: [4], 1438: [12], 1497: [11], 1878:
[6], 1911: [6], 2139: [2], 2568: [9], 2638: [4], 2663: [1], 3027: [10], 3039: [2], 3068: [8], 3216: [8], 3229: [7]}]
terribl [7, {0: [1], 131: [2], 510: [6], 1856: [4], 2043: [6], 2675: [10], 2969: [3]}]
car [43, {0: [2], 32: [3], 78: [9], 79: [7], 88: [6], 89: [6], 230: [0], 527: [10], 676: [3], 738: [12], 765: [1], 775: [5], 787: [1],
789: [3], 795: [3], 803: [3], 814: [1], 1062: [1], 1088: [9], 1152: [2], 1240: [7], 1422: [2], 1480: [3], 1486: [8], 1491: [3], 1655:
[8], 1657: [8], 1659: [8], 1849: [4], 2366: [7], 2682: [10], 2802: [8], 2928: [2], 2956: [5], 3125: [7], 3208: [8], 3212: [2], 3219:
[4], 3243: [7], 3257: [2]}]
crash [67, {0: [3], 75: [7], 84: [8], 775: [6], 776: [6], 777: [7], 778: [7], 779: [3], 780: [0], 782: [4], 783: [5], 784: [1], 785:
[3], 786: [5], 787: [4], 788: [4], 789: [4], 790: [5], 791: [0], 792: [3], 793: [3], 794: [1], 795: [0], 796: [3], 797: [13], 798: [4],
799: [1], 800: [4], 801: [5], 802: [6], 803: [5], 804: [3], 805: [6], 806: [11], 807: [9], 1041: [4], 1538: [5], 2045: [4], 2105: [7],
2111: [1], 2116: [1], 2484: [4], 2485: [3], 2486: [5], 2488: [8], 2489: [3], 2491: [4], 2498: [3], 2499: [3], 2501: [3], 2502: [4],
2503: [8], 2638: [6], 2716: [6], 2717: [6], 2718: [6], 2719: [9], 2720: [3], 2723: [2], 2724: [6], 2726: [10], 2728: [4], 2729: [7],
2814: [6]}]
heard [21, {1: [0], 145: [1], 747: [3], 954: [11], 2022: [4], 2023: [4], 2024: [3], 2025: [4], 2026: [3], 2027: [4], 2028: [4], 2029:
[3], 2030: [3], 2031: [0], 2036: [4], 2107: [12], 2216: [0], 2476: [3], 2755: [11], 2908: [4]}]
```