

UE19CS332 : Algorithms for Web and Information Retrieval
ASSIGNMENT - 1

Problem definition:

- Build a search engine for any 3 corpora of your choice,
- Your Code should be able to: Search for the terms in the query – Create Postings list
- Fill the Inverted Index
- Retrieve the data from the dictionary – Query response time.

TEAM MEMBERS :

NAME	SRN	SECTION
Priya Mohata	PES2UG19CS301	E
R Sharmila	PES2UG19CS309	E
Ritik	PES2UG19CS332	E

CORPUS – 1 : *FINANCIAL SENTIMENT ANALYSIS CORPUS*

DATASET LOCATION :

<https://drive.google.com/file/d/1ers7qOtpwLMUmM99YRf9uTZYDP6YdNRk/view?usp=sharing>

NOTEBOOK NAME : A3_P1_TEAM-20.ipynb

Link to notebook :

https://colab.research.google.com/drive/1QRe456C7_BfwLM0_KZ9maj1bqIPb9J25?usp=sharing

STEPS :

Importing all libraries

```
✓ [1] # Assignment - 1
15
# PRIYA MOHATA - PES2UG19CS301
# R SHARMILA - PES2UG19CS309
# RITIK - PES2UG19CS332

# Financial Sentiment Analysis

import pandas as pd
import numpy as np
import nltk
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
nltk.download('punkt')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
True
```

Case folding

```
# CASE FOLDING

train_data=pd.read_csv('/content/drive/MyDrive/DATASETS-AIWR/data.csv')
train_data["Sentence"] = train_data["Sentence"].str.lower()
train_data.head()
```

	Sentence	Sentiment
0	the geosolutions technology will leverage bene...	positive
1	\$esi on lows, down \$1.50 to \$2.50 bk a real po...	negative
2	for the last quarter of 2010 , componenta 's n...	positive
3	according to the finnish-russian chamber of co...	neutral
4	the swedish buyout firm has sold its remaining...	neutral

```
[3] train_data.shape

(5842, 2)
```

Renaming Columns

```
[4] train_data.rename(columns = {'Sentence':'text'}, inplace = True)
```

```
[5] train_data.columns

Index(['text', 'Sentiment'], dtype='object')
```

Sentence Tokenization

```
[6] # SENTENCE TOKENIZATION
df=train_data['text']
l=list()
for line in df:
    token=sent_tokenize(line)
    l.append(token)
```

```
[7] df=train_data['text']
train_data['sent_token']=l
```

```
train_data.head()
```

	text	Sentiment	sent_token
0	the geosolutions technology will leverage bene...	positive	[the geosolutions technology will leverage ben...
1	\$esi on lows, down \$1.50 to \$2.50 bk a real po...	negative	[\$esi on lows, down \$1.50 to \$2.50 bk a real p...
2	for the last quarter of 2010 , componenta 's n...	positive	[(for the last quarter of 2010 , componenta 's ...
3	according to the finnish-russian chamber of co...	neutral	[according to the finnish-russian chamber of c...
4	the swedish buyout firm has sold its remaining...	neutral	[the swedish buyout firm has sold its remainin...

Word Tokenization

```
[9] # WORD TOKENIZATION
df=train_data['text']
l1=list()
for line in df:
    tokens=word_tokenize(line)
    l1.append(tokens)

[10] df=train_data['text']
train_data['word_token']=l1

[11] train_data.head()
```

	text	Sentiment	sent_token	word_token
0	the geosolutions technology will leverage bene...	positive	[the geosolutions technology will leverage ben...	[the, geosolutions, technology, will, leverage...
1	\$esi on lows, down \$1.50 to \$2.50 bk a real po...	negative	[\$esi on lows, down \$1.50 to \$2.50 bk a real p...	[\$, esi, on, lows, ,, down, \$, 1.50, to, \$, 2....
2	for the last quarter of 2010 , componenta 's n...	positive	[for the last quarter of 2010 , componenta 's ...	[for, the, last, quarter, of, 2010, ,, compone...
3	according to the finnish-russian chamber of co...	neutral	[according to the finnish-russian chamber of c...	[according, to, the, finnish-russian, chamber,...
4	the swedish buyout firm has sold its remaining...	neutral	[the swedish buyout firm has sold its remainin...	[the, swedish, buyout, firm, has, sold, its, r...

Stop Words Removal

```
[12] # STOP WORDS REMOVAL
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stoplist= stopwords.words('english')

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.

[13] stoplist=set(stoplist)
l2=list()
for i in l1:
    output = [w for w in i if not w in stoplist]
    l2.append(output)
train_data['stop_words_removed']=l2

[14] train_data.head()
```

	text	Sentiment	sent_token	word_token	stop_words_removed
0	the geosolutions technology will leverage bene...	positive	[the geosolutions technology will leverage ben...	[the, geosolutions, technology, will, leverage...	[geosolutions, technology, leverage, benefon, ...
1	\$esi on lows, down \$1.50 to \$2.50 bk a real po...	negative	[\$esi on lows, down \$1.50 to \$2.50 bk a real p...	[\$, esi, on, lows, ,, down, \$, 1.50, to, \$, 2....	[\$, esi, lows, ,, \$, 1.50, \$, 2.50, bk, real, ...
2	for the last quarter of 2010 , componenta 's n...	positive	[for the last quarter of 2010 , componenta 's ...	[for, the, last, quarter, of, 2010, ,, compone...	[last, quarter, 2010, ,, componenta, 's, net, ...
3	according to the finnish-russian chamber of co...	neutral	[according to the finnish-russian chamber of c...	[according, to, the, finnish-russian, chamber,...	[according, finnish-russian, chamber, commerce...
4	the swedish buyout firm has sold its remaining...	neutral	[the swedish buyout firm has sold its remainin...	[the, swedish, buyout, firm, has, sold, its, r...	[swedish, buyout, firm, sold, remaining, 22.4, ...

3s completed at 4:27 PM

Stemming

```
[15] # 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])
```

[['geosolut', 'technolog', 'leverag', 'benefon', 'gp', 'solut', 'provid', 'locat', 'base', 'search', 'technolog', 'commun', 'platform', 'locat', 'relev', 'mi

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

	text	Sentiment	sent_token	word_token	stop_words_removed	stemmed_words
0	the geosolutions technology will leverage bene...	positive	[the geosolutions technology will leverage ben...	[the, geosolutions, technology, will, leverage...	[geosolutions, technology, leverage, benefon, ...	[geosolut, technolog, leverag, benefon, gp, so...
1	\$esi on lows, down \$1.50 to \$2.50 bk a real po...	negative	[\$esi on lows, down \$1.50 to \$2.50 bk a real p...	[\$, esi, on, lows, ,, down, \$, 1.50, to, \$, 2...	[\$, esi, lows, ,, \$, 1.50, \$, 2.50, bk, real, ...	[esi, low, bk, real, possibl]
2	for the last quarter of 2010 , componenta 's n...	positive	[for the last quarter of 2010 , componenta 's ...	[for, the, last, quarter, of, 2010, ,, compone...	[last, quarter, 2010, ,, componenta, 's, net, ...	[last, quarter, 2010, componenta, net, sale, d...
3	according to the finnish-russian chamber of co...	neutral	[according to the finnish-russian chamber of c...	[according, to, the, finnish-russian, chamber,...	[according, finnish-russian, chamber, commerce...	[accord, chamber, commerc, major, construct, c...
4	the swedish buyout firm has sold its remaining...	neutral	[the swedish buyout firm has sold its remainin...	[the, swedish, buyout, firm, has, sold, its, r...	[swedish, buyout, firm, sold, remaining, 22.4,...	[swedish, buyout, firm, sold, remain, percent,...

Lemmatization

```
# 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])
```

[nltk_data] Downloading package wordnet to /root/nltk_data...

[nltk_data] Unzipping corpora/wordnet.zip.

[['geosolutions', 'technology', 'leverage', 'benefon', 'gps', 'solution', 'providing', 'location', 'based', 'search', 'technology', 'community', 'platform',

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

	text	Sentiment	sent_token	word_token	stop_words_removed	stemmed_words	lemmatized_words
0	the geosolutions technology will leverage bene...	positive	[the geosolutions technology will leverage ben...	[the, geosolutions, technology, will, leverage...	[geosolutions, technology, leverage, benefon, ...	[geosolut, technolog, leverag, benefon, gp, so...	[geosolutions, technology, leverage, benefon, ...
1	\$esi on lows, down \$1.50 to \$2.50 bk a real po...	negative	[\$esi on lows, down \$1.50 to \$2.50 bk a real p...	[\$, esi, on, lows, ,, down, \$, 1.50, to, \$, 2...	[\$, esi, lows, ,, \$, 1.50, \$, 2.50, bk, real, ...	[esi, low, bk, real, possibl]	[esi, low, bk, real, possibility]
2	for the last quarter of 2010 , componenta 's n...	positive	[for the last quarter of 2010 , componenta 's ...	[for, the, last, quarter, of, 2010, ,, compone...	[last, quarter, 2010, ,, componenta, 's, net, ...	[last, quarter, 2010, componenta, net, sale, d...	[last, quarter, 2010, componenta, net, sale, d...
3	according to the finnish-russian chamber of co...	neutral	[according to the finnish-russian chamber of c...	[according, to, the, finnish-russian, chamber,...	[according, finnish-russian, chamber, commerce...	[accord, chamber, commerc, major, construct, c...	[according, chamber, commerce, major, construc...
4	the swedish buyout firm has sold its remaining...	neutral	[the swedish buyout firm has sold its remainin...	[the, swedish, buyout, firm, has, sold, its, r...	[swedish, buyout, firm, sold, remaining, 22.4,...	[swedish, buyout, firm, sold, remain, percent,...	[swedish, buyout, firm, sold, remaining, perce...

Building Inverted Index

```

# GENERATING INVERTED
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)

j=0
for i in final_train:
    print(i,":",final_train[i])
    if j==20:
        break;
    j=j+1

final_train1=sorted(final_train.items())

geosolut : [0, 412]
technolog : [0, 59, 109, 137, 300, 412, 427, 436, 473, 558, 657, 672, 679, 682, 712, 720, 757, 758, 823, 849, 941, 990, 1018, 1105, 1124, 1215, 1329, 1462, :
leverag : [0, 858, 3958]
benefon : [0, 300, 1053, 1312, 3808, 4158, 4993, 5604]
gp : [0, 300, 412, 3967, 4158]
solut : [0, 62, 73, 82, 85, 105, 171, 199, 201, 214, 319, 328, 341, 348, 366, 459, 496, 527, 530, 546, 607, 629, 642, 679, 687, 706, 739, 740, 773, 787, 817, :
provid : [0, 73, 82, 99, 135, 180, 191, 287, 321, 327, 357, 408, 412, 420, 436, 459, 506, 546, 607, 679, 687, 773, 820, 821, 832, 953, 970, 985, 986, 1104, :
locat : [0, 68, 187, 300, 533, 824, 983, 1454, 1977, 2038, 2137, 2164, 2230, 2284, 2807, 3006, 3131, 3153, 3248, 3393, 3605, 3625, 3680, 3808, 3828, 3966, 41 :
base : [0, 22, 198, 199, 253, 348, 352, 354, 391, 443, 524, 629, 777, 918, 981, 990, 1021, 1079, 1101, 1123, 1131, 1140, 1209, 1490, 1515, 1568, 1638, 1686, :
search : [0, 2573, 3812, 4623, 4642, 5507]
commun : [0, 7, 135, 254, 287, 305, 390, 413, 427, 506, 608, 648, 696, 740, 815, 919, 930, 986, 1051, 1160, 1434, 1620, 1696, 1763, 1859, 1888, 2076, 2129, :
platform : [0, 1088, 1568, 2230, 2559, 3645, 3877, 4069, 4082, 4158, 4186, 4276, 4905]
relev : [0, 3421, 4025, 4549, 5528]
multimedia : [0, 277, 4263]
content : [0, 390, 527, 1078, 1606, 1790, 1843, 1935, 2146, 2545, 2553, 2784, 3260, 3393, 3736, 3961, 3986, 4173, 4412, 4534, 4670, 4989, 5156, 5566]
new : [0, 14, 82, 126, 144, 253, 272, 276, 287, 310, 339, 353, 354, 375, 395, 406, 412, 415, 434, 443, 464, 495, 508, 529, 582, 592, 596, 613, 639, 642, 644, :
power : [0, 117, 606, 612, 715, 887, 1420, 1486, 1638, 1722, 1746, 1795, 1827, 2236, 2298, 2379, 2415, 2603, 2652, 2691, 2735, 2828, 2891, 2903, 3144, 3336, :
commerci : [0, 82, 142, 300, 327, 363, 1138, 1431, 1704, 1832, 1842, 2015, 2040, 2059, 2096, 2385, 2753, 2825, 3192, 3197, 3230, 3834, 4017, 4674, 4864, 4908, :
model : [0, 40, 167, 253, 297, 336, 396, 672, 786, 1044, 1128, 1271, 1291, 1298, 1616, 1649, 1760, 1885, 2060, 2311, 2647, 2691, 3007, 3038, 3145, 3258, 327 :
ssi : [1]
low : [1, 173, 310, 382, 383, 416, 550, 600, 757, 769, 1029, 1107, 1462, 1465, 1552, 1602, 1843, 1961, 2066, 2357, 2398, 2425, 2449, 2614, 3200, 3289, 3303, :
time: 363 ms (started: 2022-03-27 10:53:38 +00:00)

```

Sorting the index based on terms

```

# SORTING INDEX BASED ON TERMS
final_train1

4847,
5016,
5136,
5147,
5249,
5271,
5329,
5420,
5545,
5772)),
('backburn', [5164]),
('backdrop', [4464, 4773]),
('backhaul', [1380]),
('backlog', [307, 4446, 4753]),
('backup', [3393, 4873, 5047]),
('bad', [777, 1802, 1098, 2432, 2904, 3533, 4301, 5802]),
('badli', [200, 2726]),
('bae', [345, 542, 2514, 2916]),
('baer', [521]),
('bag', [3574, 4150, 4282, 4531, 5217, 5586]),
('bagdona', [5389]),
('bahia', [849, 1404]),
('bahr', [5752]),
('bahrain', [3996]),
('bailout', [5409]),
('baird', [1383, 3559]),
('bake', [3116]),
('baker', [5320]),
('bakeri', [1688, 5315]),
('bakman', [3082]),
('bakosch', [5527]),
('balanc',
[554,
759,
1228,
1708,
1863,
1910,
2370,
2580,
3224,
3225,
3226,
3227,
3228,
3229,
3230,
3231,
3232,
3233,
3234,
3235,
3236,
3237,
3238,
3239,
3240,
3241,
3242,
3243,
3244,
3245,
3246,
3247,
3248,
3249,
3250,
3251,
3252,
3253,
3254,
3255,
3256,
3257,
3258,
3259,
3260,
3261,
3262,
3263,
3264,
3265,
3266,
3267,
3268,
3269,
3270,
3271,
3272,
3273,
3274,
3275,
3276,
3277,
3278,
3279,
3280,
3281,
3282,
3283,
3284,
3285,
3286,
3287,
3288,
3289,
3290,
3291,
3292,
3293,
3294,
3295,
3296,
3297,
3298,
3299,
3300,
3301,
3302,
3303,
3304,
3305,
3306,
3307,
3308,
3309,
3310,
3311,
3312,
3313,
3314,
3315,
3316,
3317,
3318,
3319,
3320,
3321,
3322,
3323,
3324,
3325,
3326,
3327,
3328,
3329,
3330,
3331,
3332,
3333,
3334,
3335,
3336,
3337,
3338,
3339,
3340,
3341,
3342,
3343,
3344,
3345,
3346,
3347,
3348,
3349,
3350,
3351,
3352,
3353,
3354,
3355,
3356,
3357,
3358,
3359,
3360,
3361,
3362,
3363,
3364,
3365,
3366,
3367,
3368,
3369,
3370,
3371,
3372,
3373,
3374,
3375,
3376,
3377,
3378,
3379,
3380,
3381,
3382,
3383,
3384,
3385,
3386,
3387,
3388,
3389,
3390,
3391,
3392,
3393,
3394,
3395,
3396,
3397,
3398,
3399,
3400,
3401,
3402,
3403,
3404,
3405,
3406,
3407,
3408,
3409,
3410,
3411,
3412,
3413,
3414,
3415,
3416,
3417,
3418,
3419,
3420,
3421,
3422,
3423,
3424,
3425,
3426,
3427,
3428,
3429,
3430,
3431,
3432,
3433,
3434,
3435,
3436,
3437,
3438,
3439,
3440,
3441,
3442,
3443,
3444,
3445,
3446,
3447,
3448,
3449,
3450,
3451,
3452,
3453,
3454,
3455,
3456,
3457,
3458,
3459,
3460,
3461,
3462,
3463,
3464,
3465,
3466,
3467,
3468,
3469,
3470,
3471,
3472,
3473,
3474,
3475,
3476,
3477,
3478,
3479,
3480,
3481,
3482,
3483,
3484,
3485,
3486,
3487,
3488,
3489,
3490,
3491,
3492,
3493,
3494,
3495,
3496,
3497,
3498,
3499,
3500,
3501,
3502,
3503,
3504,
3505,
3506,
3507,
3508,
3509,
3510,
3511,
3512,
3513,
3514,
3515,
3516,
3517,
3518,
3519,
3520,
3521,
3522,
3523,
3524,
3525,
3526,
3527,
3528,
3529,
3530,
3531,
3532,
3533,
3534,
3535,
3536,
3537,
3538,
3539,
3540,
3541,
3542,
3543,
3544,
3545,
3546,
3547,
3548,
3549,
3550,
3551,
3552,
3553,
3554,
3555,
3556,
3557,
3558,
3559,
3560,
3561,
3562,
3563,
3564,
3565,
3566,
3567,
3568,
3569,
3570,
3571,
3572,
3573,
3574,
3575,
3576,
3577,
3578,
3579,
3580,
3581,
3582,
3583,
3584,
3585,
3586,
3587,
3588,
3589,
3590,
3591,
3592,
3593,
3594,
3595,
3596,
3597,
3598,
3599,
3600,
3601,
3602,
3603,
3604,
3605,
3606,
3607,
3608,
3609,
3610,
3611,
3612,
3613,
3614,
3615,
3616,
3617,
3618,
3619,
3620,
3621,
3622,
3623,
3624,
3625,
3626,
3627,
3628,
3629,
3630,
3631,
3632,
3633,
3634,
3635,
3636,
3637,
3638,
3639,
3640,
3641,
3642,
3643,
3644,
3645,
3646,
3647,
3648,
3649,
3650,
3651,
3652,
3653,
3654,
3655,
3656,
3657,
3658,
3659,
3660,
3661,
3662,
3663,
3664,
3665,
3666,
3667,
3668,
3669,
3670,
3671,
3672,
3673,
3674,
3675,
3676,
3677,
3678,
3679,
3680,
3681,
3682,
3683,
3684,
3685,
3686,
3687,
3688,
3689,
3690,
3691,
3692,
3693,
3694,
3695,
3696,
3697,
3698,
3699,
3700,
3701,
3702,
3703,
3704,
3705,
3706,
3707,
3708,
3709,
3710,
3711,
3712,
3713,
3714,
3715,
3716,
3717,
3718,
3719,
3720,
3721,
3722,
3723,
3724,
3725,
3726,
3727,
3728,
3729,
3730,
3731,
3732,
3733,
3734,
3735,
3736,
3737,
3738,
3739,
3740,
3741,
3742,
3743,
3744,
3745,
3746,
3747,
3748,
3749,
3750,
3751,
3752,
3753,
3754,
3755,
3756,
3757,
3758,
3759,
3760,
3761,
3762,
3763,
3764,
3765,
3766,
3767,
3768,
3769,
3770,
3771,
3772,
3773,
3774,
3775,
3776,
3777,
3778,
3779,
3780,
3781,
3782,
3783,
3784,
3785,
3786,
3787,
3788,
3789,
3790,
3791,
3792,
3793,
3794,
3795,
3796,
3797,
3798,
3799,
3800,
3801,
3802,
3803,
3804,
3805,
3806,
3807,
3808,
3809,
3810,
3811,
3812,
3813,
3814,
3815,
3816,
3817,
3818,
3819,
3820,
3821,
3822,
3823,
3824,
3825,
3826,
3827,
3828,
3829,
3830,
3831,
3832,
3833,
3834,
3835,
3836,
3837,
3838,
3839,
3840,
3841,
3842,
3843,
3844,
3845,
3846,
3847,
3848,
3849,
3850,
3851,
3852,
3853,
3854,
3855,
3856,
3857,
3858,
3859,
3860,
3861,
3862,
3863,
3864,
3865,
3866,
3867,
3868,
3869,
3870,
3871,
3872,
3873,
3874,
3875,
3876,
3877,
3878,
3879,
3880,
3881,
3882,
3883,
3884,
3885,
3886,
3887,
3888,
3889,
3890,
3891,
3892,
3893,
3894,
3895,
3896,
3897,
3898,
3899,
3900,
3901,
3902,
3903,
3904,
3905,
3906,
3907,
3908,
3909,
3910,
3911,
3912,
3913,
3914,
3915,
3916,
3917,
3918,
3919,
3920,
3921,
3922,
3923,
3924,
3925,
3926,
3927,
3928,
3929,
3930,
3931,
3932,
3933,
3934,
3935,
3936,
3937,
3938,
3939,
3940,
3941,
3942,
3943,
3944,
3945,
3946,
3947,
3948,
3949,
3950,
3951,
3952,
3953,
3954,
3955,
3956,
3957,
3958,
3959,
3960,
3961,
3962,
3963,
3964,
3965,
3966,
3967,
3968,
3969,
3970,
3971,
3972,
3973,
3974,
3975,
3976,
3977,
3978,
3979,
3980,
3981,
3982,
3983,
3984,
3985,
3986,
3987,
3988,
3989,
3990,
3991,
3992,
3993,
3994,
3995,
3996,
3997,
3998,
3999,
4000,
4001,
4002,
4003,
4004,
4005,
4006,
4007,
4008,
4009,
4010,
4011,
4012,
4013,
4014,
4015,
4016,
4017,
4018,
4019,
4020,
4021,
4022,
4023,
4024,
4025,
4026,
4027,
4028,
4029,
4030,
4031,
4032,
4033,
4034,
4035,
4036,
4037,
4038,
4039,
4040,
4041,
4042,
4043,
4044,
4045,
4046,
4047,
4048,
4049,
4050,
4051,
4052,
4053,
4054,
4055,
4056,
4057,
4058,
4059,
4060,
4061,
4062,
4063,
4064,
4065,
4066,
4067,
4068,
4069,
4070,
4071,
4072,
4073,
4074,
4075,
4076,
4077,
4078,
4079,
4080,
4081,
4082,
4083,
4084,
4085,
4086,
4087,
4088,
4089,
4090,
4091,
4092,
4093,
4094,
4095,
4096,
4097,
4098,
4099,
4100,
4101,
4102,
4103,
4104,
4105,
4106,
4107,
4108,
4109,
4110,
4111,
4112,
4113,
4114,
4115,
4116,
4117,
4118,
4119,
4120,
4121,
4122,
4123,
4124,
4125,
4126,
4127,
4128,
4129,
4130,
4131,
4132,
4133,
4134,
4135,
4136,
4137,
4138,
4139,
4140,
4141,
4142,
4143,
4144,
4145,
4146,
4147,
4148,
4149,
4150,
4151,
4152,
4153,
4154,
4155,
4156,
4157,
4158,
4159,
4160,
4161,
4162,
4163,
4164,
4165,
4166,
4167,
4168,
4169,
4170,
4171,
4172,
4173,
4174,
4175,
4176,
4177,
4178,
4179,
4180,
4181,
4182,
4183,
4184,
4185,
4186,
4187,
4188,
4189,
4190,
4191,
4192,
4193,
4194,
4195,
4196,
4197,
4198,
4199,
4200,
4201,
4202,
4203,
4204,
4205,
4206,
4207,
4208,
4209,
4210,
4211,
4212,
4213,
4214,
4215,
4216,
4217,
4218,
4219,
4220,
4221,
4222,
4223,
4224,
4225,
4226,
4227,
4228,
4229,
4230,
4231,
4232,
4233,
4234,
4235,
4236,
4237,
4238,
4239,
4240,
4241,
4242,
4243,
4244,
4245,
4246,
4247,
4248,
4249,
4250,
4251,
4252,
4253,
4254,
4255,
4256,
4257,
4258,
4259,
4260,
4261,
4262,
4263,
4264,
4265,
4266,
4267,
4268,
4269,
4270,
4271,
4272,
4273,
4274,
4275,
4276,
4277,
4278,
4279,
4280,
4281,
4282,
4283,
4284,
4285,
4286,
4287,
4288,
4289,
4290,
4291,
4292,
4293,
4294,
4295,
4296,
4297,
4298,
4299,
4300,
4301,
4302,
4303,
4304,
4305,
4306,
4307,
4308,
4309,
4310,
4311,
4312,
4313,
4314,
4315,
4316,
4317,
4318,
4319,
4320,
4321,
4322,
4323,
4324,
4325,
4326,
4327,
4328,
4329,
4330,
4331,
4332,
4333,
4334,
4335,
4336,
4337,
4338,
4339,
4340,
4341,
4342,
4343,
4344,
4345,
4346,
4347,
4348,
4349,
4350,
4351,
4352,
4353,
4354,
4355,
4356,
4357,
4358,
4359,
4360,
4361,
4362,
4363,
4364,
4365,
4366,
4367,
4368,
4369,
4370,
4371,
4372,
4373,
4374,
4375,
4376,
4377,
4378,
4379,
4380,
4381,
4382,
4383,
4384,
4385,
4386,
4387,
4388,
4389,
4390,
4391,
4392,
4393,
4394,
4395,
4396,
4397,
4398,
4399,
4400,
4401,
4402,
4403,
4404,
4405,
4406,
4407,
4408,
4409,
4410,
4411,
4412,
4413,
4414,
4415,
4416,
4417,
4418,
4419,
4420,
4421,
4422,
4423,
4424,
4425,
4426,
4427,
4428,
4429,
4430,
4431,
4432,
4433,
4434,
4435,
4436,
4437,
4438,
4439,
4440,
4441,
4442,
4443,
4444,
4445,
4446,
4447,
4448,
4449,
4450,
4451,
4452,
4453,
4454,
4455,
4456,
4457,
4458,
4459,
4460,
4461,
4462,
4463,
4464,
4465,
4466,
4467,
4468,
4469,
4470,
4471,
4472,
4473,
4474,
4475,
4476,
4477,
4478,
4479,
4480,
4481,
4482,
4483,
4484,
4485,
4486,
4487,
4488,
4489,
4490,
4491,
4492,
4493,
4494,
4495,
4496,
4497,
4498,
4499,
4500,
4501,
4502,
4503,
4504,
4505,
4506,
4507,
4508,
4509,
4510,
4511,
4512,
4513,
4514,
4515,
4516,
4517,
4518,
4519,
4520,
4521,
4522,
4523,
4524,
4525,
4526,
4527,
4528,
4529,
4530,
4531,
4532,
4533,
4534,
4535,
4536,
4537,
4538,
4539,
4540,
4541,
4542,
4543,
4544,
4545,
4546,
4547,
4548,
4549,
4550,
4551,
4552,
4553,
4554,
4555,
4556,
4557,
4558,
4559,
4560,
4561,
4562,
4563,
4564,
4565,
4566,
4567,
4568,
4569,
4570,
4571,
4572,
4573,
4574,
4575,
4576,
4577,
4578,
4579,
4580,
4581,
4582,
4583,
4584,
4585,
4586,
4587,
4588,
4589,
4590,
4591,
4592,
4593,
4594,
4595,
4596,
4597,
4598,
4599,
4600,
4601,
4602,
4603,
4604,
4605,
4606,
4607,
4608,
4609,
4610,
4611,
4612,
4613,
4614,
4615,
4616,
4617,
4618,
4619,
4620,
4621,
4622,
4623,
4624,
4625,
4626,
4627,
4628,
4629,
4630,
4631,
4632,
4633,
4634,
4635,
4636,
4637,
4638,
4639,
4640,
4641,
4642,
4643,
4644,
4645,
4646,
4647,
4648,
4649,
4650,
4651,
4652,
4653,
4654,
4655,
4656,
4657,
4658,
4659,
4660,
4661,
4662,
4663,
4664,
4665,
4666,
4667,
4668,
4669,
4670,
4671,
4672,
4673,
4674,
4675,
4676,
4677,
4678,
4679,
4680,
4681,
4682,
4683,
4684,
4685,
4686,
4687,
4688,
4689,
4690,
4691,
4692,
4693,
4694,
4695,
4696,
4697,
4698,
4699,
4700,
4701,
4702,
4703,
4704,
4705,
4706,
4707,
4708,
4709,
4710,
4711,
4712,
4713,
4714,
4715,
4716,
4717,
4718,
4719,
4720,
4721,
4722,
4723,
4724,
4725,
4726,
4727,
4728,
4729,
4730,
4731,
4732,
4733,
4734,
4735,
4736,
4737,
4738,
4739,
4740,
4741,
4742,
4743,
4744,
4745,
4746,
4747,
4748,
4749,
4750,
4751,
4752,
4753,
4754,
4755,
4756,
4757,
4758,
4759,
4760,
4761,
4762,
4763,
4764,
4765,
4766,
4767,
4768,
4769,
4770,
4771,
4772,
4773,
4774,
4775,
4776,
4777,
4778,
4779,
4780,
4781,
4782,
4783,
4784,
4785,
4786,
4787,
4788,
4789,
4790,
4791,
4792,
4793,
4794,
4795,
4796,
4797,
4798,
4799,
48
```

```

✓ [24] !pip install ipython-autotime
%load_ext autotime

Collecting ipython-autotime
  Downloading ipython-autotime-0.3.1-py2.py3-none-any.whl (6.8 kB)
Requirement already satisfied: ipython in /usr/local/lib/python3.7/dist-packages (from ipython->ipython-autotime) (5.5.0)
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-packages (from ipython->ipython-autotime) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-packages (from ipython->ipython-autotime) (0.7.5)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.7/dist-packages (from ipython->ipython-autotime) (1.0.18)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.7/dist-packages (from ipython->ipython-autotime) (57.4.0)
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-packages (from ipython->ipython-autotime) (2.6.1)
Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-packages (from ipython->ipython-autotime) (4.8.0)
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.7/dist-packages (from ipython->ipython-autotime) (0.8.1)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.7/dist-packages (from ipython->ipython-autotime) (5.1.1)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages (from prompt-toolkit<2.0.0,>=1.0.4->ipython->ipython-autotime) (0.2.5)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from prompt-toolkit<2.0.0,>=1.0.4->ipython->ipython-autotime) (1.15.0)
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.7/dist-packages (from pexpect->ipython->ipython-autotime) (0.7.0)
Installing collected packages: ipython-autotime
Successfully installed ipython-autotime-0.3.1
time: 169 µs (started: 2022-03-27 10:50:59 +00:00)

```

Building Positional Index

```

✓ # GENERATING POSITIONAL INDEX
pos_index = {}
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:
                pos_index[term][0] = pos_index[term][0] + 1
                if fileno in pos_index[term][1]:
                    pos_index[term][1][lineno].append(pos)
                else:
                    pos_index[term][1][lineno] = [pos]
            else:
                pos_index[term] = []
                pos_index[term].append(1)
                pos_index[term].append({})
                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;
    if count<=20:
        print(i,final[i])
    else:
        break;
final1=sorted(final.items())

```

```
geosolut [2, {0: [0], 412: [0]}]
technolog [127, {0: [10], 59: [5], 109: [3], 137: [1], 300: [16], 412: [8], 427: [7], 436: [19], 473: [2], 558: [18], 657: [7], 672: [21], 679: [8], 682: [1]
leverag [3, {0: [2], 858: [13], 3958: [6]}]
benefon [9, {0: [3], 300: [1], 1053: [4], 1312: [3], 3808: [6], 4158: [0], 4993: [0], 5604: [20]}]
gp [6, {0: [4], 300: [6], 412: [7], 3967: [0], 4158: [10]}]
solut [155, {0: [5], 62: [14], 73: [1], 82: [4], 85: [3], 105: [8], 171: [0], 199: [2], 201: [6], 214: [4], 319: [8], 328: [11], 341: [3], 348: [6], 366: [3]
provid [142, {0: [6], 73: [2], 82: [5], 99: [1], 135: [8], 180: [2], 191: [8], 287: [4], 321: [0], 327: [3], 357: [3], 408: [3], 412: [2], 420: [7], 436: [1]
locat [40, {0: [13], 68: [8], 107: [3], 300: [15], 533: [3], 824: [13], 983: [0], 1454: [10], 1977: [16], 2038: [5], 2137: [1], 2164: [6], 2230: [8], 2284:
base [80, {0: [8], 22: [1], 198: [3], 199: [3], 253: [7], 348: [3], 352: [2], 354: [3], 391: [9], 443: [14], 524: [2], 629: [23], 777: [6], 918: [22], 981:
search [6, {0: [9], 2573: [5], 3812: [2], 4623: [2], 4642: [3], 5507: [4]}]
commun [78, {0: [11], 7: [1], 135: [6], 254: [8], 287: [1], 305: [3], 390: [7], 413: [15], 427: [4], 506: [3], 608: [11], 648: [13], 696: [4], 740: [8], 815
platform [14, {0: [12], 1088: [0], 1568: [6], 2230: [5], 2559: [8], 3645: [4], 3877: [5], 4069: [11], 4082: [10], 4158: [9], 4186: [21], 4276: [14], 4905: [1]
relev [6, {0: [14], 3421: [11], 4025: [0], 4549: [9], 5528: [6]}]
multimedia [3, {0: [15], 277: [2], 4263: [1]}]
content [25, {0: [16], 390: [6], 527: [12], 1078: [13], 1606: [5], 1790: [5], 1843: [4], 1935: [2], 2146: [15], 2545: [2], 2553: [3], 2784: [9], 3260: [24],
new [274, {0: [17], 14: [6], 82: [7], 126: [2], 144: [3], 253: [0], 272: [7], 276: [2], 287: [11], 310: [2], 339: [0], 353: [0], 354: [12], 375: [6], 395: [1]
power [49, {0: [18], 117: [9], 606: [1], 612: [10], 715: [5], 887: [7], 1420: [10], 1486: [3], 1638: [9], 1722: [9], 1746: [7], 1795: [6], 1827: [1], 2236: [1]
commerci [32, {0: [19], 82: [18], 142: [3], 300: [18], 327: [14], 363: [14], 1138: [2], 1431: [1], 1704: [9], 1832: [13], 1842: [9], 2015: [8], 2040: [6], 21
model [45, {0: [20], 40: [6], 167: [2], 253: [11], 297: [1], 336: [6], 396: [24], 672: [9], 786: [5], 1044: [7], 1128: [2], 1271: [9], 1291: [2], 1298: [2],
esi [1, {1: [0]}]
time: 407 ms (started: 2022-03-27 10:55:25 +00:00)
```

```
# SORTING BASED ON THE TERMS
final1
```

Performing Boolean Queries

```
[25] # Boolean Query
# AND
def and_query(l1, l2):
    p1 = 0
    p2 = 0
    result = list()
    while p1 < len(l1) and p2 < len(l2):
        if l1[p1] == l2[p2]:
            result.append(l1[p1])
            p1 += 1
            p2 += 1
        elif l1[p1] > l2[p2]:
            p2 += 1
        else:
            p1 += 1
    return result

time: 5.6 ms (started: 2022-03-27 10:51:03 +00:00)
```

```
[26] def or_query(l1,l2):
    result=list()
    p1=0
    p2=0
    while p1 < len(l1) and p2 < len(l2):
        if l1[p1] == l2[p2]:
            result.append(l1[p1])
            p1 += 1
            p2 += 1
        elif l1[p1] > l2[p2]:
            result.append(l2[p2])
            p2 += 1
        else:
            result.append(l1[p1])
            p1 += 1
    while(p1 < len(l1)):
        result.append(l1[p1])
        p1 += 1
    while p2 < len(l2):
        result.append(l2[p2])
        p2 += 1
    return result

time: 20.9 ms (started: 2022-03-27 10:51:07 +00:00)
```

```
[29] # PERFORMING THE BOOLEAN QUERY
print("Enter the first input word : ")
input1=input()
print("Enter the second input word : ")
input2=input()

Enter the first input word :
new
Enter the second input word :
content
time: 5.14 s (started: 2022-03-27 10:52:15 +00:00)
```

```
[33] l1=final_train[input1]
l2=final_train[input2]
print("posting list for",input1 ,l1)
print("posting list for",input2,l2)
print("Resultant list: ",and_query(l1,l2))

posting list for new [0, 14, 82, 126, 144, 253, 272, 276, 287, 310, 339, 353, 354, 375, 395, 406, 412, 415, 434, 443, 464, 495, 508, 529, 582, 592, 596, 613,
posting list for content [0, 390, 527, 1078, 1606, 1790, 1843, 1935, 2146, 2545, 2553, 2784, 3260, 3393, 3736, 3961, 3986, 4173, 4412, 4534, 4670, 4989, 5151]
Resultant list: [0, 2146, 4534, 5156]
time: 5.62 ms (started: 2022-03-27 10:53:51 +00:00)
```



```

[34] print("Resultant list: ",or_query(l1,l2))
print("Length of posting list for",input1 ,len(l1))
print("Length of posting list for",input2,len(l2))
print("Length of and list: ",len(and_query(l1,l2)))
print("Resultant list :",or_query(l1,l2))
print("Length of the OR list:",len(or_query(l1,l2)))

Resultant list: [0, 14, 82, 126, 144, 253, 272, 276, 287, 310, 339, 353, 354, 375, 390, 395, 406, 412, 415, 434, 443, 464, 495, 508, 527, 529, 582, 592, 594]
Length of posting list for new 261
Length of posting list for content 24
Length of and list: 4
Resultant list : [0, 14, 82, 126, 144, 253, 272, 276, 287, 310, 339, 353, 354, 375, 390, 395, 406, 412, 415, 434, 443, 464, 495, 508, 527, 529, 582, 592, 594]
Length of the OR list: 281
time: 10.9 ms (started: 2022-03-27 10:53:57 +00:00)

[35] print("Enter the third input word : ")
input3=input()
print("Enter the fourth input word : ")
input4=input()
l3=final_train[input3]
l4=final_train[input4]
resultant=or_query(or_query(and_query(l1,l4),l3),l2)
print("Result:",resultant)
print("Result length:",len(resultant))

Enter the third input word :
model
Enter the fourth input word :
multimedia
Result: [0, 40, 167, 253, 297, 336, 390, 396, 527, 672, 786, 1044, 1078, 1128, 1271, 1291, 1298, 1606, 1616, 1649, 1760, 1790, 1843, 1885, 1935, 2060, 2146, 2146]
Result length: 66
time: 51.8 s (started: 2022-03-27 10:53:59 +00:00)

[36] resultant=and_query(or_query(l1,l3),or_query(l2,l4))
print("Result:",resultant)
print("Result length:",len(resultant))

Result: [0, 2146, 4534, 5156]
Result length: 4
time: 5.24 ms (started: 2022-03-27 10:54:54 +00:00)

```

Performing Phrase Query on Inverted Index

```

[39] # PHRASE QUERY on Inverted Index :
def phrase_query(phr):
    query=phr.split()
    for i in range(0,len(query)-1,2):
        result=and_query(final_train[query[i]],final_train[query[i+1]])
        print(result)
    print("Enter your query")
    q=input()
    phrase_query(q)

Enter your query
new content
[0, 2146, 4534, 5156]
time: 7.26 s (started: 2022-03-27 10:56:21 +00:00)

[40] print("Enter your query")
q=input()
phrase_query(q)

Enter your query
new multimedia content
[0]
time: 9.21 s (started: 2022-03-27 10:56:29 +00:00)

```

Performing Phrase Query on Positional Index

```
✓ [42] # Phrase query on positional index :  
0s  
def fetch_list(d):  
    l=list();  
    d1=d[1];  
    for i in d1:  
        l.append(i)  
    return l;  
def post_phrase_query(phr):  
    query=phr.split()  
    for i in range(0,len(query)-1,2):  
        l1=fetch_list(final[query[i]])  
        l2=fetch_list(final[query[i+1]])  
        result=and_query(l1,l2)  
        print(result)  
  
time: 6.46 ms (started: 2022-03-27 10:57:08 +00:00)
```

```
✓ [43] print("Enter your query")  
5s  
q=input()  
post_phrase_query(q)  
  
Enter your query  
new multimedia content  
[0]  
time: 5.64 s (started: 2022-03-27 10:57:11 +00:00)
```

```
✓ [44] print("Enter your query")  
4s  
q=input()  
post_phrase_query(q)  
  
Enter your query  
new content  
[0, 2146, 4534, 5156]  
time: 3.78 s (started: 2022-03-27 10:57:19 +00:00)
```