ALGI KANAR

CMPE 493 Assignment 2 REPORT

(i) Describe the steps you have performed for data preprocessing

For the data pre-process, the first thing I did was to extract each NewId, Title and Body information according to the tags indicated in the articles. For each iteration, if there is not any title or body content, I took them as empty. However, if they have any one of them then I look for any punctuations in the texts of title and body. If there is a punctuation, I replaced it white space. After that before doing the tokenization which involves dividing each word according to whitespaces, I performed case folding by lowering all the letters. Then, with regex formula I performed the tokenization. This happened for both titles and bodies for each article. When I have separate arrays for title and body, I performed stopword removal by iterating each one of them to get rid of unwanted words. After I got the filtered arrays, I concatenate them and push them into a new array which will be given to the methods used for to create inverted-index and Trie structure.

(ii) Describe the data structures that you used for representing the inverted index

To create an inverted index, I used dictionary list data structure and basically pushed this dictionary after filling it with the data obtained from pre-processing along with the Idlist (Idlist is for us to know how many iterations needs to be done. It is consisting of NewIds) Keys are the words and values are the lists with filled NewIds. By creating this kind of structure, I can easily access the occurrences in query-processing.

(iii) Describe the trie data structure that you used in your code and provide your well commented trie code in the report.

The Trie data structure I used, basically creates connected trienodes with each other according to the character they contain. After each keyword comes for insertion, it starts from the root trienode to look for each index characters of the keyword. If there is no word contains the character that it creates a new trienode. However, before moving forward to the new trienodes, it records the newly created trienode to the child list of current nodes. This happens because when a new word comes it will know that this character has been found before so it will branch according to that and move forward directly without creating a new trienode. This process of insertion will be kept for each word until they come to an end.

Trie class also includes depthfirstsearch, fetchedpossiblewords methods which are used in the queryprocess to obtain the possible words starting with certain pieces. (I also did the necessary commenting for these methods. You can see it in below code.)

```
class TrieNode:
    #This class will create the each trie node objects

    def __init__(self, character):
        # the character stored in this node
        self.char = character

        # whether this can be the end of a word
        self.is_end = False

        # a dictionary for all the child trienodes( keys are characters, values
are trienodes )
        self.children = {}

class Trie():
    #Trie object

    def __init__(self):
        #Trienode is the starting node
        #it won't store any character
        self.root = TrieNode("")

    #used method to fill trie
    def insert(self, keyword):
        """Insert a word into the trie"""
        trienode = self.root

        # Going through the all characters in the keyword
        for character in keyword:
            # Look if there is a child,move to that trienode
            if character in trienode.children:
                trienode = trienode.children[character]
            else:
                # If a character is not found,form a newtrienode in the trie
                newtrienode = TrieNode(character)
                trienode.children[character] = newtrienode
                trienode = newtrienode

        # Mark the end of the each keyword
        trienode.is_end = True

    # Below methods (fetchpossiblewords,dfs) are designed for the outside access
from query.py
    #in order to achieve queryprocess they will traverse the Trie object.

    def depthfirstsearch(self, trienode, pref):
        #depth first search algorithm
        #it will go down until see the end trienode to extract the words.

        if trienode.is_end:
            self.output.append((pref + trienode.char))

        for child in trienode.children.values():
            self.depthfirstsearch (child, pref + trienode.char)
```

```
    def fetchpossiblewords(self, wordprefix):

        # Use an array within the class to keep all possible outputs
        # because for each prefix there can be many combinations
        self.output = []
        trienode = self.root

        # Check if the wordprefix is in the trie
        for char in wordprefix:
            if char in trienode.children:
                trienode = trienode.children[char]
            else:
                # if can’t find the wordprefix, return empty list
                return []

        # Traverse the trie to get all candidates by the dept first search
algorithm
        #This will fill up the all possible outputs.
        self.depthfirstsearch(trienode, wordprefix[:-1])

        # Sort the all possible result outputs
        return self.output
```

QUERY EXAMPLES

tur*

```
C:\Users\Algi>cd desktop

C:\Users\Algi\Desktop>python prep.py

C:\Users\Algi\Desktop>query.py
Please enter query key: *
Please don't forget to enter query key: tur*
[28, 45, 178, 223, 248, 276, 295, 331, 334, 335, 342, 372, 389, 444, 627, 635, 652, 730, 748, 809, 835, 862, 894, 899, 920, 922, 936, 1019, 1074, 1081, 1097, 1252, 1254, 1478, 1480, 1498, 1539, 1579, 1580, 1611,
 1685, 1696, 1749, 1808, 1867, 1911, 1917, 1927, 1935, 1959, 1967, 1975, 2001, 2025, 2034, 2084, 2137, 2190, 2223, 2246, 2318, 2400, 2475, 2495, 2514, 2517, 2522, 2552, 2662, 2666, 2758, 2759, 2765, 2768, 2900,
2960, 2961, 2962, 2963, 2968, 2970, 3004, 3076, 3090, 3188, 3217, 3269, 3273, 3432, 3434, 3449, 3454, 3456, 3462, 3463, 3468, 3480, 3505, 3517, 3518, 3528, 3539, 3563, 3597, 3613, 3635, 3640, 3701, 3745, 3759, 3
792, 3798, 3818, 3837, 3841, 3938, 3966, 4002, 4017, 4026, 4045, 4057, 4081, 4085, 4089, 4092, 4094, 4100, 4106, 4108, 4123, 4132, 4137, 4196, 4202, 4223, 4226, 4266, 4286, 4328, 4470, 4562, 4574, 4610, 4625, 46
36, 4647, 4649, 4661, 4662, 4691, 4692, 4713, 4727, 4736, 4755, 4756, 4776, 4809, 4860, 4897, 4916, 4931, 4946, 5022, 5038, 5130, 5150, 5157, 5163, 5171, 5180, 5205, 5206, 5210, 5219, 5221, 5225, 5230, 5244, 524
8, 5250, 5257, 5264, 5265, 5267, 5274, 5278, 5376, 5386, 5391, 5394, 5395, 5410, 5432, 5456, 5514, 5515, 5527, 5561, 5574, 5630, 5655, 5656, 5715, 5719, 5726, 5761, 5769, 5787, 5879, 5888, 5891, 5978, 5985, 6112
, 6280, 6328, 6335, 6346, 6349, 6358, 6363, 6372, 6395, 6400, 6472, 6481, 6521, 6633, 6706, 6722, 6870, 6896, 6935, 6949, 6972, 6999, 7003, 7032, 7051, 7075, 7102, 7103, 7105, 7155, 7254, 7280, 7319, 7366, 7540,
 7552, 7578, 7593, 7649, 7654, 7740, 7757, 7797, 7811, 7825, 7871, 7892, 7934, 7951, 7966, 8029, 8038, 8085, 8087, 8110, 8125, 8136, 8155, 8178, 8191, 8192, 8193, 8434, 8435, 8439, 8441, 8510, 8578, 8611, 8614,
8615, 8618, 8619, 8627, 8644, 8671, 8743, 8839, 8884, 8902, 8905, 8944, 8961, 8963, 8980, 9048, 9096, 9127, 9139, 9158, 9160, 9175, 9177, 9180, 9187, 9189, 9229, 9244, 9248, 9302, 9374, 9537, 9603, 9638, 9658, 9
662, 9706, 9707, 9728, 9734, 9739, 9741, 9759, 9760, 9761, 9763, 9769, 9778, 9784, 9819, 9831, 9848, 9908, 9926, 9969, 9974, 9977, 10080, 10106, 10111, 10123, 10133, 10168, 10175, 10194, 10228, 10252, 10255, 102
78, 10280, 10282, 10317, 10325, 10328, 10348, 10367, 10395, 10433, 10452, 10511, 10522, 10539, 10548, 10588, 10595, 10605, 10611, 10612, 10617, 10620, 10621, 10627, 10630, 10641, 10642, 10648, 10667, 10668, 1068
8, 10694, 10699, 10700, 10703, 10715, 10739, 10743, 10744, 10778, 10797, 10862, 10863, 10905, 10910, 10915, 11056, 11118, 11124, 11167, 11186, 11205, 11221, 11227, 11243, 11252, 11254, 11279, 11404, 11413, 11472
, 11512, 11535, 11654, 11661, 11714, 11763, 11768, 11773, 11781, 11793, 11795, 11797, 11810, 11823, 11827, 11830, 11833, 11837, 11840, 11854, 11870, 11885, 11887, 11894, 11918, 11944, 11987, 12123, 12135, 12180,
 12292, 12311, 12320, 12328, 12330, 12349, 12396, 12410, 12432, 12463, 12479, 12484, 12522, 12524, 12598, 12701, 12709, 12720, 12759, 12799, 12806, 12815, 12848, 12877, 12878, 12905, 12909, 12917, 12943, 12948,
12983, 12994, 13012, 13052, 13073, 13123, 13129, 13179, 13244, 13250, 13251, 13259, 13314, 13331, 13350, 13379, 13406, 13419, 13458, 13501, 13561, 13606, 13629, 13662, 13689, 13752, 13807, 13832, 13907, 13919, 1
4005, 14051, 14068, 14132, 14190, 14196, 14266, 14279, 14287, 14419, 14454, 14588, 14609, 14698, 14749, 14799, 14839, 14857, 14860, 14862, 14863, 14869, 14874, 14880, 14881, 14890, 14912, 14925, 14930, 15014, 15
063, 15085, 15194, 15200, 15254, 15338, 15369, 15372, 15374, 15380, 15381, 15386, 15387, 15390, 15397, 15421, 15427, 15438, 15551, 15556, 15639, 15766, 15781, 15884, 15894, 15957, 16009, 16028, 16072, 16086, 161
13, 16116, 16137, 16140, 16141, 16159, 16175, 16181, 16191, 16207, 16217, 16220, 16239, 16268, 16310, 16322, 16352, 16427, 16491, 16504, 16540, 16556, 16602, 16636, 16669, 16736, 16750, 16756, 16773, 16785, 1693
5, 16948, 16953, 16957, 16961, 17012, 17018, 17058, 17083, 17103, 17148, 17170, 17173, 17201, 17203, 17218, 17243, 17248, 17250, 17252, 17264, 17296, 17305, 17306, 17325, 17329, 17368, 17372, 17396, 17402, 17433
, 17474, 17508, 17549, 17553, 17693, 17700, 17731, 17737, 17779, 17810, 17839, 17862, 17884, 17886, 17891, 17896, 17900, 17901, 17915, 17921, 17922, 17937, 17962, 18034, 18061, 18085, 18102, 18138, 18143, 18174,
 18231, 18288, 18293, 18350, 18399, 18438, 18467, 18469, 18483, 18495, 18497, 18507, 18514, 18563, 18570, 18676, 18709, 18776, 18796, 18823, 18855, 18888, 18917, 18934, 18990, 18994, 19028, 19038, 19039, 19068,
19087, 19157, 19250, 19285, 19353, 19378, 19385, 19436, 19476, 19483, 19494, 19499, 19532, 19559, 19561, 19565, 19566, 19584, 19588, 19627, 19672, 19740, 19750, 19786, 19797, 19836, 19884, 19909, 19968, 19976, 2
0029, 20048, 20076, 20102, 20204, 20259, 20389, 20485, 20492, 20496, 20614, 20632, 20635, 20636, 20683, 20703, 20705, 20760, 20764, 20772, 20781, 20823, 20860, 20867, 20868, 20870, 20897, 20940, 21202, 21340, 21
392, 21500, 21503, 21508, 21536, 21561, 21565, 21577]
```

tur

```
C:\Users\Algi\Desktop>python query.py
Please enter query key: tur
[809]
```

people

```
C:\Users\Algi\Desktop>query.py
Please enter query key: people
[28, 54, 178, 225, 269, 331, 340, 364, 367, 375, 540, 597, 714, 867, 878, 889, 894, 895, 955, 990, 1026, 1300, 1332, 1408, 1617, 1725, 1827, 1895, 1896, 1906, 1908, 1956, 2000, 2168, 2184, 2267, 2305, 2452, 2485
, 2502, 2777, 2799, 2813, 2819, 2822, 2824, 2844, 2853, 2944, 2955, 2957, 2958, 2959, 2965, 2967, 2972, 2981, 3047, 3112, 3148, 3195, 3265, 3279, 3302, 3332, 3366, 3388, 3403, 3442, 3445, 3488, 3493, 3532, 3534,
 3535, 3537, 3551, 3563, 3568, 3576, 3635, 3650, 3717, 3762, 3790, 3940, 3984, 4003, 4037, 4040, 4110, 4126, 4209, 4233, 4290, 4299, 4303, 4356, 4382, 4443, 4625, 4678, 4727, 4764, 4890, 4940, 5063, 5127, 5137,
5145, 5158, 5190, 5206, 5224, 5262, 5268, 5289, 5298, 5345, 5376, 5486, 5638, 5739, 5800, 5829, 5920, 6112, 6117, 6126, 6150, 6250, 6338, 6344, 6346, 6351, 6382, 6395, 6472, 6603, 6838, 6844, 6899, 7022, 7070, 7
088, 7105, 7149, 7161, 7290, 7393, 7405, 7461, 7545, 7549, 7566, 7605, 7658, 7792, 7892, 7918, 7933, 8006, 8024, 8038, 8068, 8069, 8072, 8080, 8089, 8150, 8158, 8160, 8181, 8198, 8242, 8290, 8384, 8439, 8452, 84
76, 8583, 8592, 8595, 8626, 8645, 8660, 8662, 8679, 8800, 8943, 8944, 8991, 9024, 9046, 9110, 9140, 9153, 9156, 9179, 9180, 9183, 9194, 9221, 9226, 9254, 9256, 9327, 9370, 9408, 9445, 9512, 9536, 9581, 9602, 960
7, 9691, 9696, 9708, 9713, 9717, 9722, 9732, 9758, 9774, 9780, 9834, 9978, 10036, 10070, 10118, 10191, 10239, 10286, 10306, 10342, 10486, 10622, 10635, 10666, 10667, 10689, 10771, 10996, 11014, 11029, 11074, 111
60, 11172, 11175, 11178, 11198, 11204, 11215, 11275, 11426, 11442, 11492, 11624, 11654, 11688, 11740, 11766, 11768, 11769, 11785, 11793, 11886, 11991, 12194, 12233, 12272, 12277, 12431, 12440, 12507, 12533, 1261
2, 12700, 12718, 12743, 12750, 12827, 12876, 12879, 12887, 12907, 12909, 13053, 13068, 13080, 13123, 13215, 13248, 13249, 13283, 13331, 13541, 13586, 13592, 13609, 13622, 13629, 13633, 13652, 13662, 13949, 14063
, 14390, 14419, 14630, 14635, 14654, 14691, 14710, 14753, 14767, 14854, 14864, 14889, 14890, 14891, 14936, 14982, 15033, 15073, 15091, 15149, 15224, 15369, 15372, 15392, 15417, 15434, 15442, 15443, 15485, 15674,
 15742, 15781, 15824, 15932, 16044, 16061, 16064, 16082, 16089, 16093, 16100, 16137, 16158, 16161, 16181, 16352, 16402, 16421, 16503, 16558, 16562, 16636, 16731, 16756, 16785, 16829, 16860, 16864, 16906, 16959,
16960, 16997, 17084, 17103, 17166, 17190, 17201, 17318, 17348, 17384, 17415, 17509, 17521, 17532, 17636, 17657, 17676, 17750, 17760, 17839, 17862, 17879, 17884, 17890, 17915, 17934, 17935, 17959, 18043, 18049, 1
8228, 18267, 18359, 18383, 18442, 18552, 18616, 18878, 18923, 19039, 19043, 19064, 19126, 19127, 19157, 19158, 19191, 19276, 19289, 19292, 19316, 19392, 19401, 19411, 19415, 19483, 19583, 19673, 19746, 19816, 19
877, 19930, 19964, 19986, 20028, 20051, 20232, 20437, 20481, 20571, 20614, 20705, 20764, 20782, 20823, 20860, 20867, 20870, 20897, 21149, 21235, 21340, 21422, 21525, 21575, 21577]
```