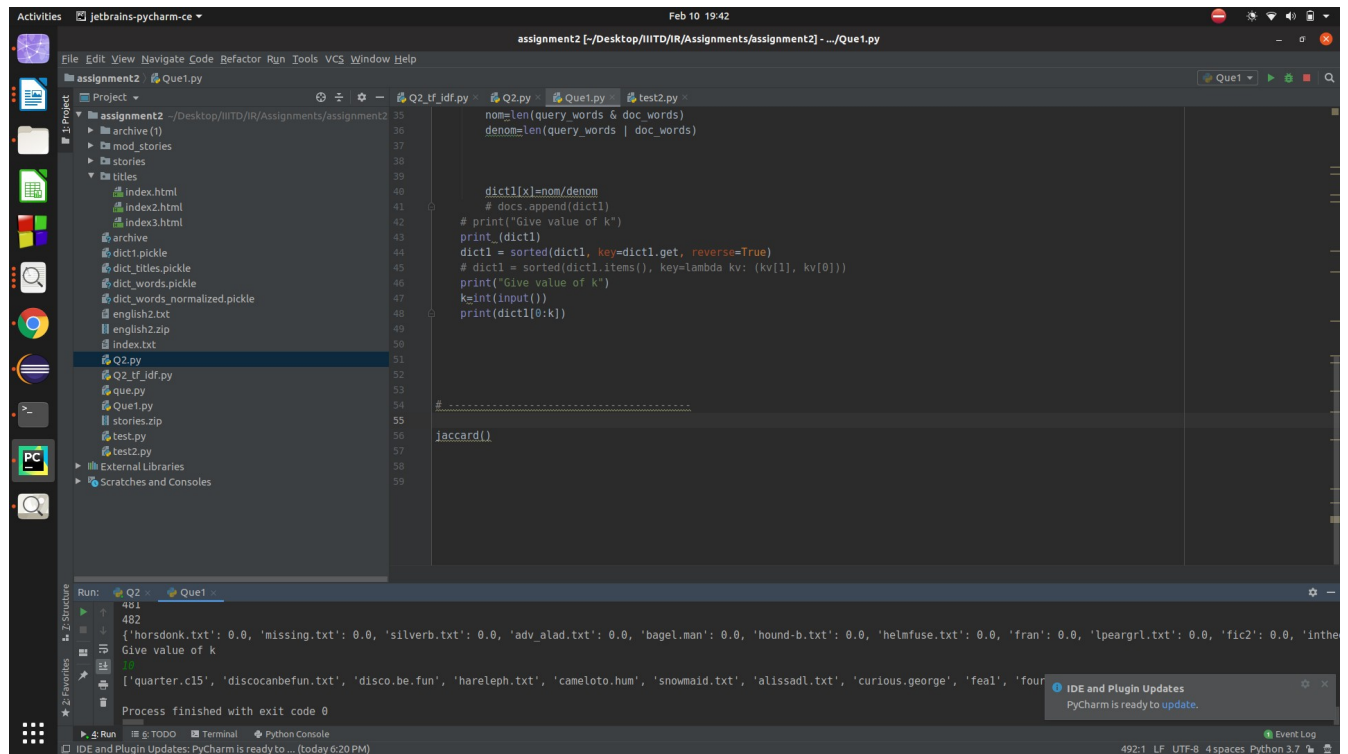


Analysis

Q(i).

1). Output:



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script named `Que1.py` with the following code:

```
33 nom=len(query_words & doc_words)
34 denom=len(query_words | doc_words)
35
36
37
38
39
40 dict1[x]=nom/denom
41 # docs.append(dict1)
42 # print("Give value of k")
43 print(dict1)
44 dict1 = sorted(dict1, key=dict1.get, reverse=True)
45 # dict1 = sorted(dict1.items(), key=lambda kv: (kv[1], kv[0]))
46 print("Give value of k")
47 k=int(input())
48 print(dict1[0:k])
49
50
51
52
53
54 # .....
55
56 jaccard()
57
58
59
```

The Run console at the bottom shows the output of the script:

```
481
482
483 {'horsdonk.txt': 0.0, 'missing.txt': 0.0, 'silverb.txt': 0.0, 'adv_alad.txt': 0.0, 'bagel.man': 0.0, 'hound-b.txt': 0.0, 'helmfuse.txt': 0.0, 'fran': 0.0, 'lpearl.txt': 0.0, 'fic2': 0.0, 'inthe
Give value of k
484
485 ['quarter.c15', 'discocanbefun.txt', 'disco.be.fun', 'hareleph.txt', 'cameloto.hum', 'snowmaid.txt', 'alissadl.txt', 'curious.george', 'feal', 'four
Process finished with exit code 0
```

A notification at the bottom right states: "IDE and Plugin Updates: PyCharm is ready to update."

This technique is biased towards the small documents but is fastest among the given techniques for the entire process. One more disadvantage is that we take more time after query entering in this technique.

2) Tf-Idf based document retrieval:

Output:

```

159 # for x in dict1.keys():
160 #     p=cal_tf_idf(dict1,x,n)
161 #     # print(p)
162 #
163 #     dict1[x].append(p)
164 #     for i in range(n):
165 #         dict1[x][i]=dict1[x][i]*dict1[x][-1]
166 #
167 # with open('dict words.pickle', 'wb') as handle:
168 #     pickle.dump(dict1, handle, protocol=pickle.HIGHEST_PROTOCOL)
169 with open('dict words.pickle', 'rb') as handle:
170     dict1 = pickle.load(handle)
171 with open('dict words.normalized.pickle', 'wb') as handle:
172     pickle.dump(dict1, handle, protocol=pickle.HIGHEST_PROTOCOL)
173
174 tf_idf()

```

```

Run: Q2 - Q2_tf_idf
/usr/bin/python3.7 /home/gaurav/Desktop/IIITD/IR/Assignments/assignment2/Q2_tf_idf.py
-----Enter Query-----
10
Finding Docs.....
-----Enter value of k-----
10
['batlsalau.txt', 'dakota.txt', 'day.in.mcdonald', 'mcdonaldl.txt', 'gulliver.txt', 'vgilante.txt', 'sick-kid.txt', 'keepmodu.txt', 'hitch2.txt', 'dskool.txt']
-----Enter Query-----
disco can be fun
Finding Docs.....
-----Enter value of k-----
10
['dakota.txt', 'radar_ra.txt', 'cow.exploder', 'quest', 'rocket.sf', 'kzap.txt', 'panama.txt', 'kharian.txt', 'arctic.txt', 'keepmodu.txt']
-----Enter Query-----
disco can be fun
Finding Docs.....
-----Enter value of k-----
10
['foosee.txt', 'disco.be.fun', 'discocanbefun.txt', 'chik', 'cybersla.txt', 'robotech', 'hitch2.txt', 'cooldark.txt', 'cooldark.sto', 'hitch3.txt']
-----Enter Query-----

```

In tf-idf based retrieval we don't consider tf of the query. But slightly fast than vector-based approach

3) Tf-Idf based vector space document retrieval:

```

231 for x in query_words2:
232     # print(x)
233     # print(len(dict1[x]))
234     # print(dict2[x][-1])
235     list4.append(dict2[x][-1])
236 # print(list4)
237 list5 = length_Normalize(list4)
238 for x in list1:
239     dict3[x]=score_cos_Sim(dict2,i_query_words2,list5,list1)#####
240     # dict3[x] = score_tf_idf(dict2, i, query_words2, list1)#####
241     i = i + 1
242 # print(dict3)
243 dict3 = sorted(dict3, key=dict3.get, reverse=True)
244 # print(list1)
245 print("-----Enter value of k-----")
246 k = int(input())

```

```

tf_idf() while (1) : for x in list1

```

```

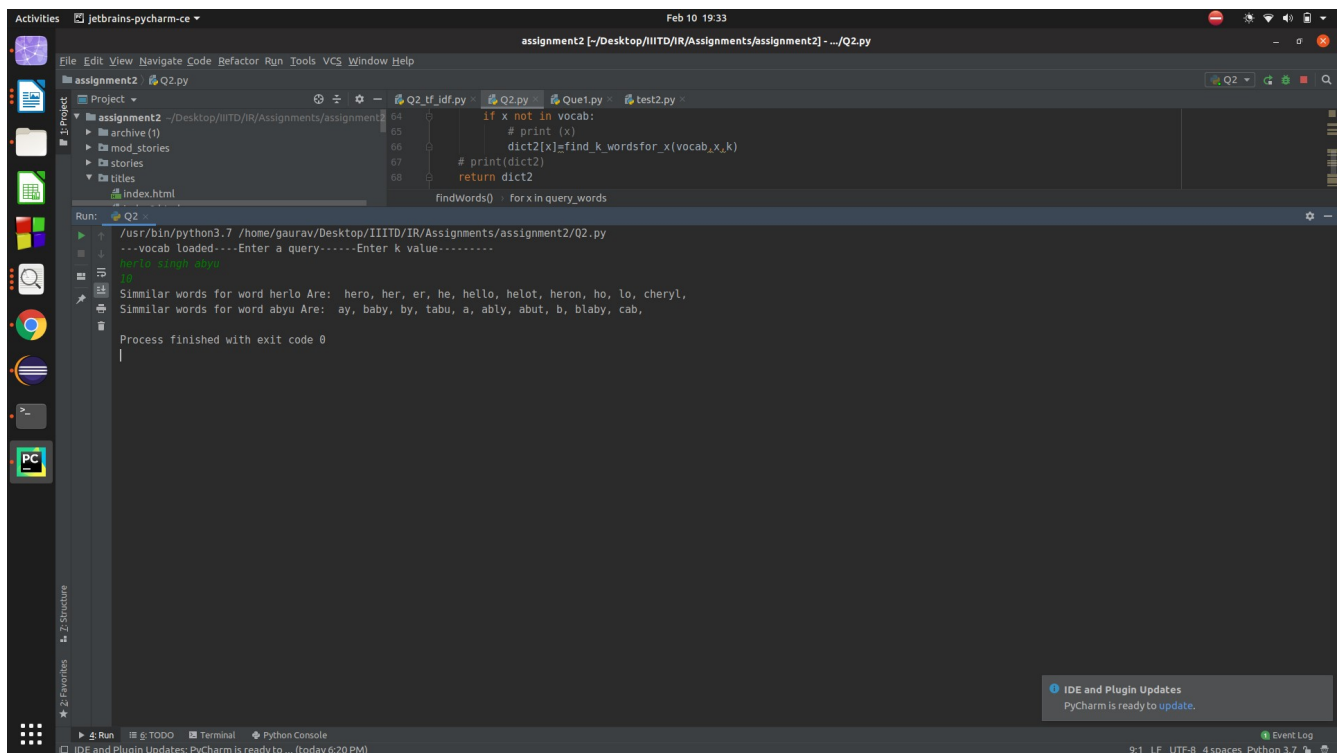
Run: Q2 - Q2_tf_idf
/usr/bin/python3.7 /home/gaurav/Desktop/IIITD/IR/Assignments/assignment2/Q2_tf_idf.py
-----Enter Query-----
disco can be fun
Finding Docs.....
-----Enter value of k-----
10
['foosee.txt', 'disco.be.fun', 'discocanbefun.txt', 'chik', 'robotech', 'cybersla.txt', 'hitch2.txt', 'cooldark.txt', 'cooldark.sto', 'hitch3.txt']
-----Enter Query-----
disco can be fun
Finding Docs.....
-----Enter value of k-----
10
['dakota.txt', 'radar_ra.txt', 'cow.exploder', 'quest', 'rocket.sf', 'kzap.txt', 'panama.txt', 'kharian.txt', 'arctic.txt', 'keepmodu.txt']
-----Enter Query-----

```

In Tf-Idf based vector space document retrieval we consider the frequency of the term.

Variant comparison: I took 2 variants for determining tf in which variant2 is biased towards tf in the document so by using variant1 we can resolve this problem.

Q(ii):



```
assignment2 [~/Desktop/IIITD/IR/Assignments/assignment2] - .../Q2.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project
assignment2
  archive(1)
  mod_stories
  stories
  titles
  index.html
  findWords() for x in query_words
Q2.py
Q2.py
Q2.py
test2.py
Run:
Q2
/usr/bin/python3.7 /home/gaurav/Desktop/IIITD/IR/Assignments/assignment2/Q2.py
---vocab loaded---Enter a query-----Enter k value-----
herlo 221gh abyu
Similar words for word herlo Are: herlo, her, er, he, hello, helot, heron, ho, lo, cheryl,
Similar words for word abyu Are: ay, baby, by, tabu, a, ably, abut, b, blaby, cab,
Process finished with exit code 0
IDE and Plugin Updates: PyCharm is ready to update.
```