Ahmet Atakan Bozkurt 02.23.2020

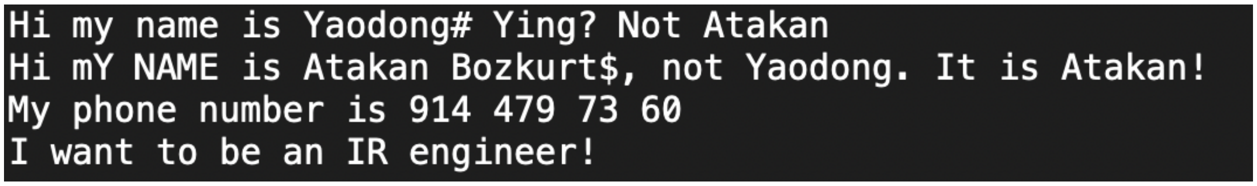
Yaodong Ying

Information Retrieval Project Part 1

In this project, we used Python programming language to build a dictionary for the terms exist in our input file “200_title.txt” and our test case “test.txt”.

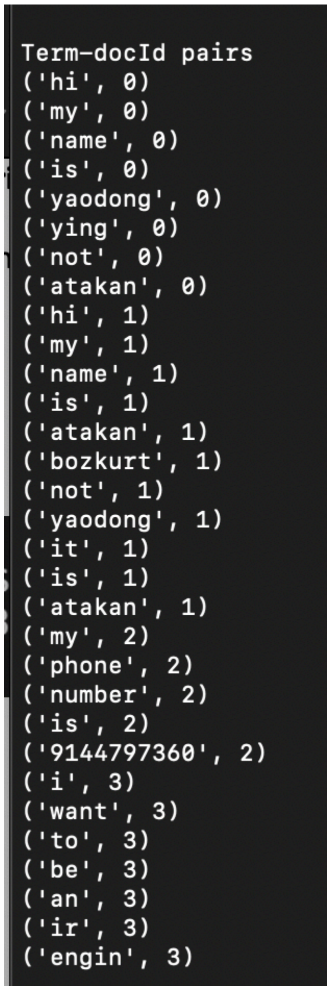
Comprehensive test case:

Input file:



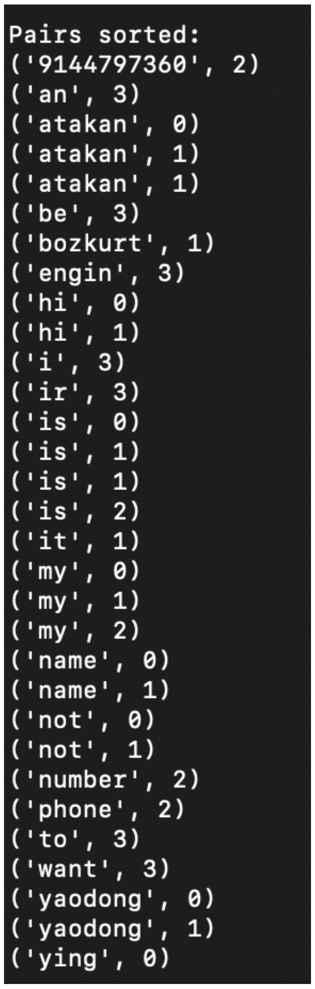
```
Hi my name is Yaodong# Ying? Not Atakan
Hi mY NAME is Atakan Bozkurt$, not Yaodong. It is Atakan!
My phone number is 914 479 73 60
I want to be an IR engineer!
```

term-docId pairs



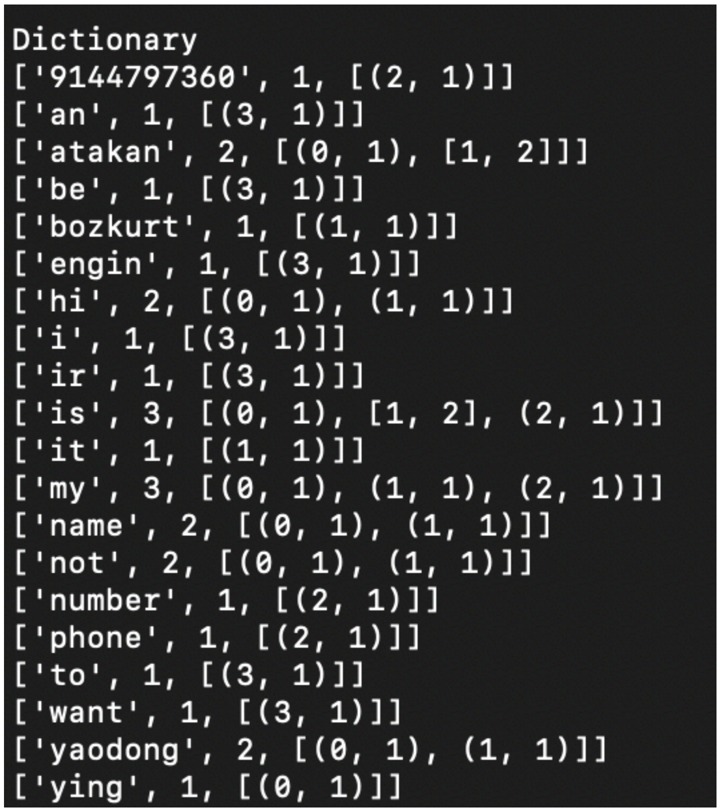
```
Term-docId pairs
('hi', 0)
('my', 0)
('name', 0)
('is', 0)
('yaodong', 0)
('ying', 0)
('not', 0)
('atakan', 0)
('hi', 1)
('my', 1)
('name', 1)
('is', 1)
('atakan', 1)
('bozkurt', 1)
('not', 1)
('yaodong', 1)
('it', 1)
('is', 1)
('atakan', 1)
('my', 2)
('phone', 2)
('number', 2)
('is', 2)
('9144797360', 2)
('i', 3)
('want', 3)
('to', 3)
('be', 3)
('an', 3)
('ir', 3)
('engin', 3)
```

term-docId after sorting



```
Pairs sorted:
('9144797360', 2)
('an', 3)
('atakan', 0)
('atakan', 1)
('atakan', 1)
('be', 3)
('bozkurt', 1)
('engin', 3)
('hi', 0)
('hi', 1)
('i', 3)
('ir', 3)
('is', 0)
('is', 1)
('is', 1)
('is', 2)
('it', 1)
('my', 0)
('my', 1)
('my', 2)
('name', 0)
('name', 1)
('not', 0)
('not', 1)
('number', 2)
('phone', 2)
('to', 3)
('want', 3)
('yaodong', 0)
('yaodong', 1)
('ying', 0)
```

Dictionary in format (‘term’, document frequency , postings list)
Postings list in format (document id , term frequency)



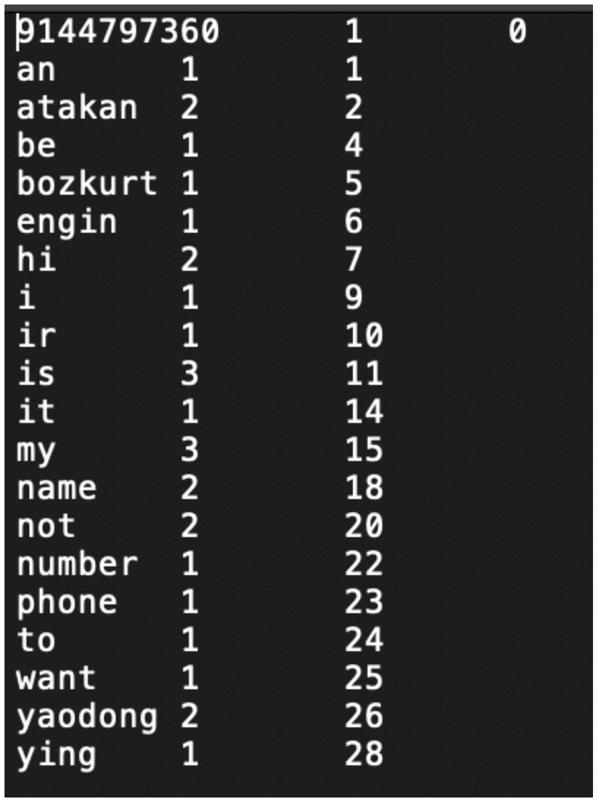
```
Dictionary
['9144797360', 1, [(2, 1)]]
['an', 1, [(3, 1)]]
['atakan', 2, [(0, 1), [1, 2]]]
['be', 1, [(3, 1)]]
['bozkurt', 1, [(1, 1)]]
['engin', 1, [(3, 1)]]
['hi', 2, [(0, 1), (1, 1)]]
['i', 1, [(3, 1)]]
['ir', 1, [(3, 1)]]
['is', 3, [(0, 1), [1, 2], (2, 1)]]
['it', 1, [(1, 1)]]
['my', 3, [(0, 1), (1, 1), (2, 1)]]
['name', 2, [(0, 1), (1, 1)]]
['not', 2, [(0, 1), (1, 1)]]
['number', 1, [(2, 1)]]
['phone', 1, [(2, 1)]]
['to', 1, [(3, 1)]]
['want', 1, [(3, 1)]]
['yaodong', 2, [(0, 1), (1, 1)]]
['ying', 1, [(0, 1)]]
```

Results:

- Special characters are successfully removed and white spaces between numbers are removed. Query “9144797360” proves this requirement. term-docId pairs are generated
- “is” appears in 3 different documents, thus document frequency is 3.
- Pairs are sorted in alphabetic order depending on term(numbers are treated as strings)
- “Atakan” appears in 2 different documents, once in first document and twice in second document. This query proves that multiple occurrence of a term in same document is successfully stored in postings list with the corresponding document Id.
- Every posting list in dictionary were sorted depending on docId.
- High recall application. We return every possible document that we can at the moment.

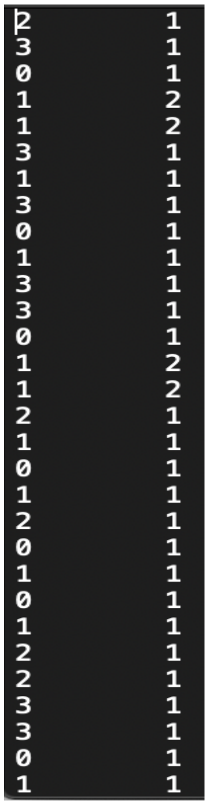
Output files

dictionary



```
9144797360        1        0
an       1        1
atakan   2        2
be       1        4
bozkurt  1        5
engin    1        6
hi       2        7
i        1        9
ir       1        10
is       3        11
it       1        14
my       3        15
name     2        18
not      2        20
number   1        22
phone    1        23
to       1        24
want     1        25
yaodong  2        26
ying     1        28
```

postings list



```
2        1
3        1
0        1
1        2
1        2
3        1
1        1
3        1
0        1
1        1
3        1
3        1
0        1
1        2
1        2
2        1
1        1
0        1
1        1
2        1
0        1
1        1
0        1
1        1
2        1
2        1
3        1
3        1
0        1
1        1
```

To run test case commandline usage is : python3 Proj1.py ./test.txt

To run 200_title.txt usage is : python3 Proj1.py ./200_title.txt