Problem on Anagrams

John Goldsmith

Spring 2020

The goal of this problems set is to use a sorted list of the letters of a string in order to quickly find interesting relations among words. This is not a difficult problem, and it is more of an exercise.

We will call a list of symbols that appear in alphabetized order an *alphabetized word*. This is a convenient way to represent *multisets*, also known as *bags of letters*. Multisets differ from sets in that an element may appear more than once in a multiset. If we have two multiset A = {a,b,c} and B={b,c,d}, the union of A and B, $A \cup B$, is the multiset {a,b,b,c,c,d}. The intersection of two alphabetized words is the longest common subsequence shared by them, and we indicate this $A \cap B$. We define $C - B$ as the (unique) multiset A such that C is the union of A and B.

Write a Python program called "anagrams.py" that will do the following (and note that it should take seconds and not minutes to run):

1. Read in a list of words (one word per line) from a file with at "txt" extension, or in a ".dx1" format.[1] Set a minimum word length of 8. If there are any upper case letters, make them lower case (obviously, there would be nothing useful about these words if you did not make them lower case).
2. For each word, sort the letters alphabetically, and keep a dict with the sorted letters as key, and as its value a list of the words which provided that bag of letters.
3. If w is a key of a dict, and the length of its value is greater than 1, then we have found an anagram. We define the length of such a list as the anagram's *size*. All of the words on such a list are of the same length, and we call that length the *length* of the anagram.
4. **(10 points:)** Use the wordlist for English with 235,000 entries which you will find on the canvas site with this problem.
   - All anagrams (=list of words) should be alphabetized.
   - Give all sets of anagrams whose length is 8 or greater, sorted by size, ordered by increasing size.
   - At the beginning of each section of a given size, print "Anagrams of size [whatever]".
   - Within anagrams of the same size, sort by length, longest last. At the beginning of each section of a given size, print "Anagrams of length [whatever]".
   - Among those of the same size and length, alphabetize by first word.
   - Print a table whose columns are size, whose rows are length, and whose entries are the number of anagrams.

	2	Clear README file
	2	Alphabetizing each word
Points	2	Adding each word to dict with alphabetized word as key
	2	Sorting output as requested
	2	Create appropriate output report

On a super-large corpus, some of the output looked like this:

[1]A dx1 file has 1 word per line, followed optionally by an integer representing the words count in some corpus; lines that begin with # are just commment lines.

```
recommednation recommendation recommnedation
conservation's conversation's conversations'
greenish-yellow yellow-greenish yellowish-green
admininstration admninistration adniminstration
psychoanalyst's psychoanalysts' pyschoanalyst's
adiministration adminisitration administratiion
unconsitutional unconstituional unconstiutional
enterpreneurial entrepreneurial entreprenuerial
parliamentarins parliamentrians parlimentarians
creutzfeld-jakob jakob-creutzfeld kreutzfeld-jacob
administration's administrations' adminsitration's
cold-temperature computer-altered computer-related
peitermaritzburg pietermaritzburg pietermartizburg
director-generals directors-general general-creditors
...
hettinger retighten tethering tightener
license's licenses' silence's silences'
armstrong granstrom rangstrom strongarm
creditor's creditors' director's directors'
bertelsman bertlesman lambertsen resemblant
invaluable unavailble unavilable unvailable
dawn-hotel hand-towel now-halted two-handle
arts-often faster-not front-seat no-fatters
hitters-mo short-item short-time time-short
and-carpet carpet-and part-dance tap-dancer
first-hole life-short rifle-shot short-life
hits-three hitters-he theirs-the there-this
anti-three in-theater retina-the theater-in
alerations rationales realisaton senatorial
re-animator retro-mania rome-tirana tirana-rome
gone-native negative-on no-negative vintage-one
englander's englanders' glenarden's greenland's
conservation converations conversation converstaion
center-stage centre-stage secret-agent stage-center
dealership's dealerships' leadership's leaderships'
commisioners commissioenr commissioner recommission
```

5. **Here is a thought-provoking question, just for fun:** can you implement a function that will select only those anagram sets that are really interesting and surprising? Don't ask what "interesting and surprising" means: that's part of the problem.

   Here are some especially interesting anagrams (in my opinion) from English:

Anagrams pairs	
licenses	silences
algorithm	logarithm
cautioned	education
continued	unnoticed
generates	teenagers
grandiose	organised
integrals	triangles
percussion	supersonic
striptease	tapestries
colonialist	oscillation
entirety	eternity