HUNGRY SCANNER

Sunday, 14.03.20201

—

Problem Statement

A string is composed of words defined as continuous runs of alphanumeric characters separated by separators (spaces, commas, periods, semi colons, exclamation marks, any other punctuation symbol **except apostrophes)**. The scanner counts words, collecting those together where a common substring of length 4 or greater occurs.

Example 1:

The hungry scanner keeps a suspicious watch on doctors and their unsuspecting patients.

In the given sentence, suspicious and unsuspecting have a common substring of length 4 "susp".

Thus the scanner would output something like this :

The : 1
hungry: 1

scanner: 1
keeps: 1
a: 1
suspicious, unsuspecting: 2
watch: 1
on: 1
doctors: 1
and: 1
their: 1
patients: 1

Example 2:
Don't know how much wood would a woodchuck chuck if A woodchuck could chuck wood, he won't chuck it all?

a:1
all:1
he:1
how:1
if:1
it:1
wood, **woodchuck**: 2
would, **could**: 2
Don't, **won't**: 2
know: 1
much: 1
chuck: 1

Design an algorithm that can work on a very long string stored in a file, and output a table as above in a file.

Each set of common-substring words should be listed on a line, along with a count of their joint occurrence.

Don’t load the entire file in memory to conserve system resources!

Input Format

File contains several paragraphs of texts. The text is separated by connectors (spaces, line breaks, commas, periods, semi colons, exclamation marks, any other punctuation symbol **except apostrophes**), which are not to be included as part of a word and should not be used to find similar substrings with other words (except apostrophes).

Output Format

Words in a combination (comma separated) : Count of words in the combination.

For eg. jackie : 1

clever, ever : 2

tonight, might, eight : 3

Save the output in a file, which will be used to generate a score later.

Scoring

10 points are given for each word you count.

If a combination is found then for that combination the score will be:

(10 + (count of words-1)*20) + Bonus

Bonus points are given on the basis of how good the combination is. One failure situation is unrelated words with common substrings. For example, consider the following words:

Assignment, Amendment, Assigned

Clearly the first two words have a common root. Unfortunately, the simple approach also identifies **Assignment** and **Amendment** as having a common root due to the presence of the string "**ment**". But, we have a situation where "**Assignment**" and "**Assigned**" could be counted in two slots.

Find an approach to "break ties" in these cases. What logic can you apply to declare that [Assignment, Assigned] is a better match than [Assignment, Amendment] ?

0 points are given for a combination, if there is a word which has already been used earlier.