CS2002301 & EC2002302 Data Structures

Homework #5

Due Date: 2021/1/11 23:55

TA email: michael071020@gmail.com

Kaggle URL: https://www.kaggle.com/c/ntust-data-structures-2020-homework-5

Introduction

For the final homework, you're asked to implement an efficiency English word checker. If a word is detected to be mis-spelled, you also have to provide suggestions for the user. For example, if "structur" is a user's input, "structure" is a reasonable suggestion.

Instructions

First, please create a dictionary. In this homework, we will use the CMU pronouncing dictionary. We have preprocessed the dictionary and uploaded it to Kaggle; please download the dictionary on "data" tab in the competition page.

Note that the structure of the dictionary is a key point in this homework. We recommend you to use "hash"; however, it's free to use other data structures. (Trees may help!)

For the suggestions to a mis-spelled word, you can create a confusable set and then delete those not in the dictionary. To be more specific, for a given mis-spelled word, the set can be generated by the following operations:

1. Insert: add a lowercase letter to all possible positions in the word.
   For instance:
   `ace → {Xace|X=a~z}∪{aXce|X=a~z}∪{acXe|X=a~z}∪{aceX|X=a~z}`
2. Delete: delete a character in the word.
   For instance: `ace → {ac, ae, ce}`
3. Substitute: substitute a character in the word with a lowercase letter.
   For instance: `ace → {Xce|X=a~z}∪{aXe|X=a~z}∪{acX|X=a~z}`
4. Transpose: exchange two neighboring characters in the word.
   For instance: `ace → {cae, aec}`

We can denote the above set as ***S1***(string), which is set of words that have edit distance of 1 to the given string. For instance, the number of elements in ***S1***('something') is 494 (with duplicate removed).

Similarly, we can have ***S2***(string) = ***S1***(***S1***(string)), which is a set of words that have edit distance of 2 to the given string. This set is easy to write, but takes much longer to compute. For instance, the size of ***S2***('something') is 114,324 (with duplicates removed). Then we need to remove words not in the dictionary. If this step is denoted as clean, then our approach is like this:

A. x1=***S1***('something') → size(x1) = 494
B. x2=***S1***(x1) → size(x2) = 114,324
C. out=clean(x1∪x2) → size(out)=5, that is, out='seething', 'smoothing', 'something', 'somethings', 'soothing'

Input Format

There are two input files:

1. `dictionary.txt`: The dictionary. Please download it from Kaggle.
2. `test.txt`: The test cases. In the input file, each line consists a word. The words are either correct (in the dictionary) or mis-spelled (not in the dictionary). For example, A typical example of input file is as follows:

```
happy
xasperate
zynxyz
```

DO NOT change the filename. The TAs will use different files but have the same file name to test your program.

Note that for the input file, you should remove the copyright by yourself.

Output Format

The output should be in CSV (Comma-Separated Values) format.

The first column should be the original word in the input file.

The second column should be the answer (suggested words in the dictionary), separated by a space. If the given word is correct (already in the dictionary), please output "OK". If the given word is mid-spelled but we can't find any suggested in the dictionary, then print "NONE". Note that all the given word and the word in answer should be lowercase.

For instance, the output corresponding to the previous example input will be:

```
word,answer
happy,OK
xasperate,aspirate desperate exasperate exasperated
zynxyz,NONE
```

Scoring

The TAs will evaluate your score base on the accuracy and the efficiency of your program. You should also submit a report to Moodle system.

```
YOUR FINAL SCORE = 9 * accuracy * efficiency + Report
```

a. Accuracy

   We will use Kaggle website to evaluate the accuracy of your program. Please submit your CSV file to the Kaggle competition page to test the accuracy.

   The accuracy score is based on the hidden test file which contains 500 words.

b. Efficiency

   To prevent you from overfitting, we will execute your program on **100 private entries** for scoring efficiency.

   The score in this section is evaluated as follows:

```
soft_baseline, hard_baseline = 60s, 120s
if execution_time > hard_baseline:
 efficiency = 0.0
elif hard_baseline >= execution_time > soft_baseline:
 efficiency = 0.3
elif soft_baseline >= execution_time > 45 seconds:
 efficiency = 0.7
else:
 efficiency = 1.0
```

c. Report (1 point)

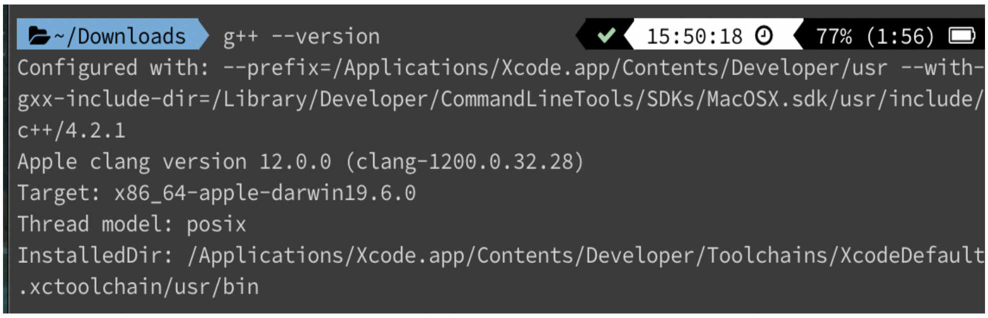
Please submit a report to Moodle system. Your report score is either 0.5 or 1. Of course, if you forget to submit the report, you won't get any point in this section.

Submission

Please submit your code and a report to Moodle. (compress in a zip file)

Note that:

1. DO NOT upload the input file.
2. DO NOT change the name of the input files. (`dictionary.txt, test.txt`)
3. If you are using Windows OS:
   Please submit a Visual Studio repo. The TAs will use VS 2019 to test your program.



```
~/Downloads  g++ --version                              ✔  15:50:18   77% (1:56)
Configured with: --prefix=/Applications/Xcode.app/Contents/Developer/usr --with-
gxx-include-dir=/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/
c++/4.2.1
Apple clang version 12.0.0 (clang-1200.0.32.28)
Target: x86_64-apple-darwin19.6.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault
.xctoolchain/usr/bin
```

4. If you are using Linux or MacOS:
   Please submit your source code. Make sure your code can be successfully compiled in g++ 12.0.
   Below is the version of g++ in TA's MacBook:

Warning

1. **Any form of cheating and plagiarism is not allowed. You will get 0 point if you are caught cheating.**
2. **Please make sure that your output satisfies the output constraint.**