EngSci Press Project Final Report

Yunhao Qian

SN: 1005684225

April 7, 2020

1 Introduction

Hello.

2 Objectives

The core dictionary program should:

1. *Launch and response fast.* A slow start-up puts the user in a bad mood even before s/he starts using the program.

   Metric: Measure the time interval between a user request and its response. Shorter time in seconds is better. Startup time should be less than 1 second.

2. *Use memory efficiently.* Users might run the program on an outdated computer or a virtual machine, which typically has very limited memory. Large memory use hurts performance and can cause system failure.

   Metric: Measure the increased memory usage after loading the same dictionary dataset. Less memory in megabytes is better.

3. *Add dictionary entries easily.* The provided data have a lot of typos. Users like me might be unsatisfied and want to customize them. After following a clear and simple procedure, users should be able to add data files with the same format.

   Metric: Count the number of operations to load a CSV file into the dictionary dataset. Fewer operations are better.

The story writer program should:

1. *Produce grammatically correct sentences.* To generate meaningful and logical stories is beyond my ability. To tell my story writer apart from a monkey hitting keys, the only way is to force my production grammatically correct.

   Metric: Copy and paste the produced text into Microsoft Word. Green underlines flag grammatical errors. Fewer grammatical errors per sentence are better.

2. *Control the length of generated text accurately.* Sentence generation is slow, so it is a waste of time to work on unneeded sentences.

Metric: Calculate the percentage difference between the user-specified length and the length of generated text. Smaller average difference is better.

3 Detailed Framework

3.1 High-Level Overview

3.2 Languages

I use C for the core dictionary because it runs faster and provides more precise memory control. I initially wrote it in Python, but it took 3 seconds to launch and violated the time constraint. The bottleneck turns out to be CPU computation as opposed to disk IO. Moving to C should effectively speed it up since compiled languages typically compute much faster than interpreted languages.

I use Python for the story writer because it is easier to code, supports regular expression and features various sampling methods. Usage of these functionality is described in <Section>. Python libraries such as NumPy have a mature and efficient C/Fortran back-end. Compared to reinvented wheels, they are faster, more robust and easier to debug. Moreover, exception mechanism in Python makes it simpler to handle special cases that appear in a natural language.

3.3 Data Structures

3.4 Software Implementation

4 Results

Hello.

5 Future Work/Conclusion

Hello.

Appendices

Hello.