CSC180 Project 3 Report

# Part A

After downloading the raw text of the novels from project Gutenberg, and running the program using the text within them, we found that the performance on test.txt was 67.5% correct (ie. The program answered 67.5% of the questions correctly). Then, we modified the text to remove the extra stuff put in by the project Gutenberg people (like licensing, credits, etc.) and ran the program again. The program now had a performance of 70.0% on test.txt. This was most likely due to the fact that the removal of the “extra” text within the project Gutenberg files resulted in the removal of many sentences which would not have been used in “normal” English conversation or vocabulary, increasing the accuracy of the program. The code used to generate the results is as follows:

d = build\_semantic\_descriptors\_from\_files(["warandpeace.txt", "swannsway.txt"])  
print(run\_similarity\_test("test.txt", d, cosine\_similarity))

# Part B

Table 1: Similarity Measure and Performance on Test

|  |  |
| --- | --- |
| Similarity Measure | Performance on Test (% Correct) |
| Cosine | 0.7 |
| Negative Euclidean Space Distance | 0.625 |
| Negative Normalized Euclidean Space Distance | 0.7 |

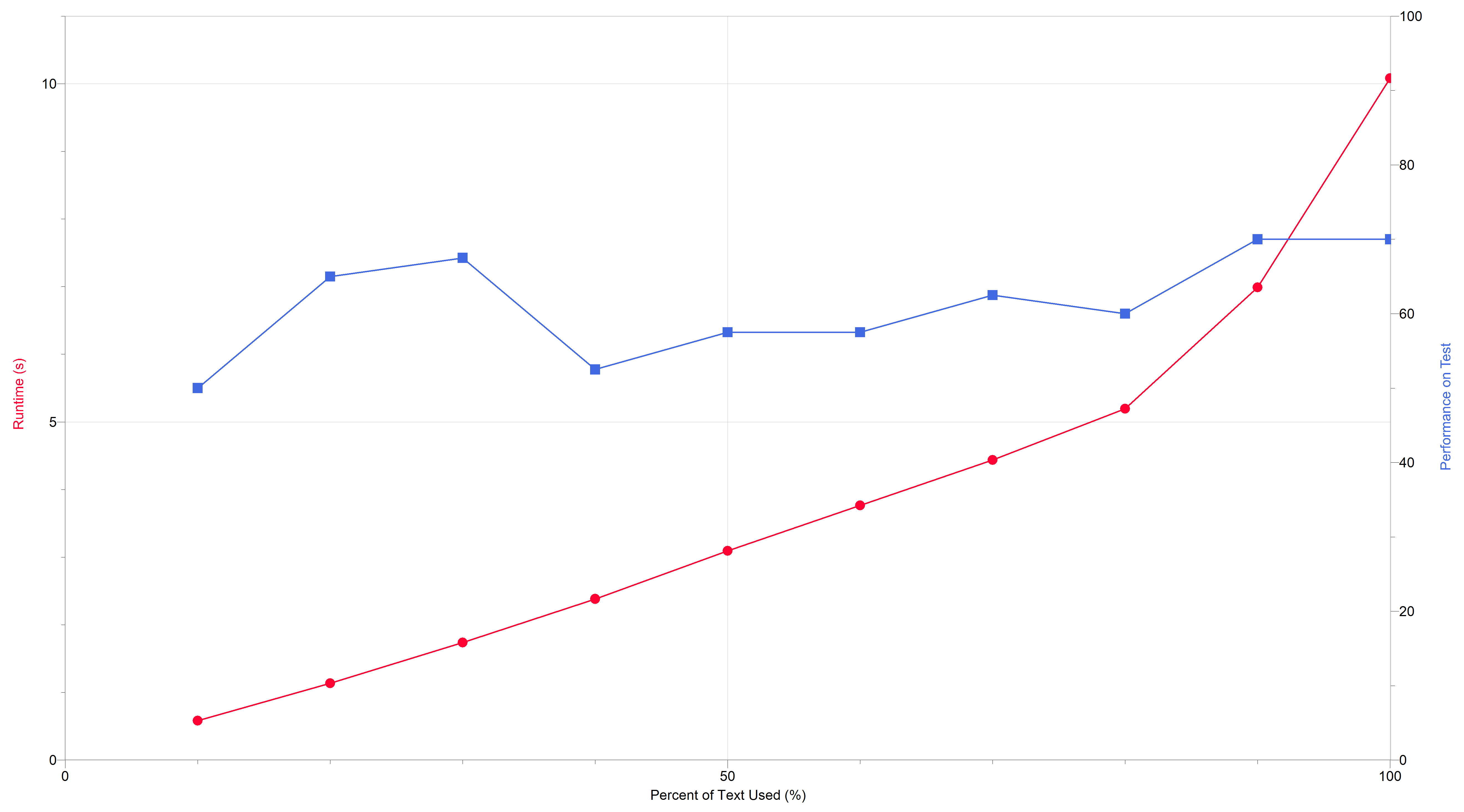
From further testing, it appears that the negative Euclidean space distance similarity measure suffers from a problem of the similarity being affected by the magnitude of the vectors (ie. A set of vectors with larger magnitude naturally returns a larger result, compared to a shorter set of vectors with higher similarity). The negative normalized Euclidean space distance similarity measure does not seem to suffer this problem.

# Part C

Table 2: Percent of Text Used, Runtime, and Performance on Test

|  |  |  |
| --- | --- | --- |
| Percent of Text Used (%) | Runtime (s) | Performance on Test (% Correct) |
| 10 | 0.582 | 50.0 |
| 20 | 1.136 | 65.0 |
| 30 | 1.738 | 67.5 |
| 40 | 2.382 | 52.5 |
| 50 | 3.092 | 57.5 |
| 60 | 3.766 | 57.5 |
| 70 | 4.436 | 62.5 |
| 80 | 5.194 | 60.0 |
| 90 | 6.988 | 70.0 |
| 100 | 10.081 | 70.0 |

Figure 1: Runtime and Performance on Test versus Percent of Text Used



It was noted that the time taken to run the program followed a somewhat exponential curve relative to the percent of text used. It was also noted that the performance on the text showed no noticeable trends.

It is very likely that the reason that the performance on test fluctuates so drastically is because the program automatically guesses the first answer when it does not find a dictionary entry for a word in the choices, thereby getting some questions correct by chance when it does not have enough data from the novels.

It was also noted that creating a new python shell before each run of the code significantly reduced the run time of the program, especially with a larger amount of text percent used. The recorded values in the table above are the lowest runtimes found (ie. Refreshing the python shell after each time the code was ran).