Reports of Phase Five

Steps to run this program:

To run the program, you can use the command:

```
py simple.py
```

Please note that for this program to run you need to have ‘files’ folder (containing all the html documents) in your project folder and you need to have beautifulsoup, scipy packages installed.

Detailed summary of the updates made to the code:

In this phase, we are tasked to find out the similarity matrix of the entire corpus and also perform agglomerative clustering. Initially I started by creating a zero-matrix named ‘num’ using numpy library. Post that I have created a list named ‘den’. In num I will storing all the numerator values that we need to construct the similarity matrix and in den we will be storing all the denominator values. I have iterated through each item in postings list and each item is nothing but the postings of a particular word in the corpus. Since each posting contain the document name and tfidf score of the word in that document, I have extracted document index from document name. I have performed the update to num matrix in such a way that num value is updated by the sum of product of tfidf scores of the current two documents. I have updated the den value by the sum of den value plus the square of current tfidf score. After iterating through all the postings in the postings list, we are left with the num matrix and den matrix with all the updated values. Using these two matrices, we find out the similarity matrix. I have done that by updating each num value by dividing it with product of square root of the den values for those respective documents. After performing this for the all the indices in the num matrix, we will end up with the similarity matrix. I have created similarity_matrix.txt to write the similarity matrix into it. Post that I have normalized all the values in the num matrix using min-max normalization. I have stored all the normalized values in normalized matrix. Post that I have created an empty dis matrix. In this dis matrix I have stored all the distance values which I have found using the formula 1 – normalized_matrix. This matrix is going to be the matrix upon which we perform the clustering. I have used a package named scipy which contains all the powerful functions that we require to perform the clustering. I have used linkage method to perform complete link clustering. I have updated the max_dist value with 0.4 so that once the maximum distance of 0.4 is reached, the clustering will stop. I have used fcluster function to give names to the cluster. I have created a for loop and every time a merge occurs, I have printed between which clusters the merge is happening. Here, since we are using distance instead of similarity score so the most similar documents are not the ones that are merged together. I have attached the first 100 lines of the output in the following pages of the report. Please note that if we merge clusters i and j then the updated cluster is going to be i.

The most similar and most dissimilar documents:

I have iterated though each value in the similarity matrix and find out the maximum similarity score and minimum similarity score and the documents between which these scores occur. Documents 492.html and 498.html are most similar documents of the entire corpus with an unnormalized similarity score of 2.23 while documents 98.html and 495.html have the least similarity score and are the most dissimilar documents of the corpus.

Data Structures Used:

I have used three different matrices to store the similarity matrix, normalized matrix, distance matrix. I have used a list to store all the denominator values that we require to compute the similarity matrix. I have used fcluster function to create all the clusters.

Complexity:

Since we are using multiple packages and performing high level computations, the complexity and run time have increased a bit from the previous phase. It takes 27 seconds for this program to run which is quite high and have many rooms for improvement. If we were to code the clustering section on our own without using any libraries, we might be able to get it down a little.

First 100 lines of output:

```
Cluster 79 merged with Cluster 71

Cluster 71 merged with Cluster 134

Cluster 134 merged with Cluster 115

Cluster 115 merged with Cluster 103

Cluster 103 merged with Cluster 138

Cluster 138 merged with Cluster 94

Cluster 94 merged with Cluster 83

Cluster 83 merged with Cluster 85

Cluster 85 merged with Cluster 79

Cluster 79 merged with Cluster 114

Cluster 114 merged with Cluster 115

Cluster 115 merged with Cluster 119

Cluster 119 merged with Cluster 138
```

Cluster 138 merged with Cluster 107

Cluster 107 merged with Cluster 83

Cluster 83 merged with Cluster 82

Cluster 82 merged with Cluster 121

Cluster 121 merged with Cluster 78

Cluster 78 merged with Cluster 113

Cluster 113 merged with Cluster 115

Cluster 115 merged with Cluster 82

Cluster 82 merged with Cluster 138

Cluster 138 merged with Cluster 107

Cluster 107 merged with Cluster 125

Cluster 125 merged with Cluster 130

Cluster 130 merged with Cluster 71

Cluster 71 merged with Cluster 79

Cluster 79 merged with Cluster 114

Cluster 114 merged with Cluster 132

Cluster 132 merged with Cluster 121

Cluster 121 merged with Cluster 107

Cluster 107 merged with Cluster 120

Cluster 120 merged with Cluster 121

Cluster 121 merged with Cluster 138

Cluster 138 merged with Cluster 128

Cluster 128 merged with Cluster 95

Cluster 95 merged with Cluster 132

Cluster 132 merged with Cluster 133

Cluster 133 merged with Cluster 108

Cluster 108 merged with Cluster 121

```
Cluster 121 merged with Cluster 130
Cluster 130 merged with Cluster 87
Cluster 87 merged with Cluster 79
Cluster 79 merged with Cluster 138
Cluster 138 merged with Cluster 132
Cluster 132 merged with Cluster 131
Cluster 131 merged with Cluster 123
Cluster 123 merged with Cluster 121
Cluster 121 merged with Cluster 130
Cluster 130 merged with Cluster 139
Cluster 139 merged with Cluster 79
Cluster 79 merged with Cluster 114
Cluster 114 merged with Cluster 81
Cluster 81 merged with Cluster 124
Cluster 124 merged with Cluster 109
Cluster 109 merged with Cluster 121
Cluster 121 merged with Cluster 85
Cluster 85 merged with Cluster 119
Cluster 119 merged with Cluster 131
Cluster 131 merged with Cluster 134
Cluster 134 merged with Cluster 120
Cluster 120 merged with Cluster 124
Cluster 124 merged with Cluster 123
Cluster 123 merged with Cluster 112
Cluster 112 merged with Cluster 85
Cluster 85 merged with Cluster 108
Cluster 108 merged with Cluster 131
```

Cluster 131 merged with Cluster 135

Cluster 135 merged with Cluster 120

Cluster 120 merged with Cluster 124

Cluster 124 merged with Cluster 107

Cluster 107 merged with Cluster 110

Cluster 110 merged with Cluster 85

Cluster 85 merged with Cluster 106

Cluster 106 merged with Cluster 131

Cluster 131 merged with Cluster 134

Cluster 134 merged with Cluster 120

Cluster 120 merged with Cluster 107

Cluster 107 merged with Cluster 111

Cluster 111 merged with Cluster 87

Cluster 87 merged with Cluster 116

Cluster 116 merged with Cluster 120

Cluster 120 merged with Cluster 114

Cluster 114 merged with Cluster 81

Cluster 81 merged with Cluster 120

Cluster 120 merged with Cluster 123

Cluster 123 merged with Cluster 138

Cluster 138 merged with Cluster 85

Cluster 85 merged with Cluster 120

Cluster 120 merged with Cluster 113

Cluster 113 merged with Cluster 133

Cluster 133 merged with Cluster 85

Cluster 85 merged with Cluster 120

Cluster 120 merged with Cluster 121

Cluster 121 merged with Cluster 130

Cluster 130 merged with Cluster 120

Cluster 120 merged with Cluster 95

Cluster 95 merged with Cluster 142

Cluster 142 merged with Cluster 120

Cluster 120 merged with Cluster 91