In my proposal, I have said that I wanted to do text summarization for the student essays that had similar topic by comparing each essay and forming a new combine essay. The main idea was to ignore all the stop words that add no value to the text, such as, won’t, we, who, and focus on useful information only. To actually do this, I compare each paragraph in one essay with another paragraph in the second essay in the same position. Then, I would give each sentence in the paragraph a weighted value and compare the sentence with each other sentence in the same position. The sentence with the higher weighted value would be added to the new paragraph.

A more detailed version would be; each paragraph would be broken down into sentences. Each sentence would be strip of their stopwords and be split up into words. Each word would then be added to the defaultdict(int) function, that would keep track of each unique words and the frequency of each unique words. The highest frequency number would be used as the denominator to divide each words. For example, the highest frequency word has a count of 150 and the second highest has a count of 100. The first word would have a value of 150/150 = 1. The second word would have a value of 100/150 = 0.667. This is the method I’ve used to give each words a value. Then I would add up all the valued words in the sentence to get a sum valued for the sentence. This is where the weighted sentence came from. After each sentences in the paragraph is weighted, then the comparison would begin.

Unfortunately, I could not achieved all the promises I’ve made in my proposal. I did not achieve the comparison of two essays and combining them into a single summarized essay. The problem was something that should have been obvious, but I did not foresee it in the beginning. The problem was that each essays have different length. The total number of each sentences and total number of each paragraphs in one essay are different from another essay. I have tried to averaged out the number of sentences in each paragraph, but my program keep giving me an error, saying that the sentence did not exist when it should have existed. I didn’t have time to try a different strategy. The good news is, I have manage to summarized one essay at a time and printed out the results that should still give a more meaningful insight into the analysis of the essay.

I will now go over the code I’ve used in my program. I’m not going to go into much details, just the overview. In my program I’ve reuse the code from my assignment 3. My assignment 3 was topic modeling. I have said that I wanted to summarize essays with similar topic. So, it made sense to reuse code. I did not achieve my goals, but the code was still used. In lines 10 - 225 of my program, it finds all the topic in the corpus. Lines 225 -229, it uses a nested for-loop to find all the essays with similar topic. Line 231 is where the code for summarization starts. The variable “tokenizer” is assigned to hold the RegexpTokenizer() function. This function will be use to remove all punctuations. Line 232, with the variable “stopWords” is an array that holds all the stopwords from importing nltk.corpus stopwords. Line 233 -247, read each essays one at a time in the corpus, split the essay into paragraphs, split each paragraph into sentences, split up each sentences into words, and used a for-loop to remove each stopwords for the important words to be placed into the defaultdict(int) function named “freqStudent” to keep of the frequency of each unique words. Lines 248 - 254, uses the variable “free” to do the frequency count. The variable “fun” holds an array of tuples and each tuples has their key element assign to a word and has their value element assign to the frequency count. The first tuple is the one highest frequency count. The variable “high” holds the tuple with the highest frequency count. Lines 251-252, extract the value of the highest frequency count and placed it into a variable named “highest.” Lines 253 - 259, has two directories named “keyDict” and “valueDict” to keep track of the position for the key and value of the tuples. For example, the first tuple will have its key element at position 0 in the “keyDict” dictionary and have its value element at the 0 position in the “valueDict” dictionary. Line 260, have a dictionary named “highDict” to hold the weighted words. Lines 261 - 272, will use the highest frequency count to divide each frequency count in the “valueDict”, then it will use the look at the position of each key in the “valueDict” to match the position of the key in the “keyDict” to find the value element, which is the word. After finding the word and assigning a value for that word, it will be placed in the “highDict” dictionary. Lines 273 - 274, sorts the “highDict” by their value from lowest to highest, and then reverse the order to make sure that the highest value will appear first. Lines 275 - 301, prints everything out.

If I was given more time to try a new strategy, I would try to figure out why my program gave me an error say that a sentence did not exist when it should have existed. I would also try to figure how to add multiple essays together. If I have three essays, I would probably add the first two essay together as one essay, and use that essay to combine the third essay. For example, something like (a + b) + c.

Now for the analysis, in my ResultPhoto1.PNG. I have printed out all the paragraphs in each essays and all the sentences with their value shown. The highest weighted sentence would appear first and then the next one. I have also printed out, tuples, the total sum value for the paragraph, the total number of sentences in the paragraph, and the average value for the paragraph. This was to give a meaningful insight into the paragraph. In this case, the first paragraph in the picture shows that it has an average value of 6.1667 and the second paragraph has an average value of 9.1666. This means that the second paragraph gives more value to the essay. We can also see that the sentences in the second paragraph have more value to them then the sentences in the first paragraph. If a student was to use this program, they could see which to change and maybe add new sentences to increase the average value of the paragraph. It is worth noting that most of the high value sentences have lots of words in them. A student could also use this to focus on adding more meaningful words to the sentence. For example, in the first paragraph, they should focus on using more “I”, “efforts”, and “donations” to make their sentences more meaningful.

Another thing worth mentioning, originally I was going to use the average value of the paragraph as the baseline to print out the sentences that had greater value or equal value to the baseline. For example, the first paragraph has an average of value of 6.1667. If I use that 6.1667 as the baseline, then the first three sentences would have been printed out. To show that the first three sentences are the most important ones in the paragraph. But, I decided against this. I want to print out all the sentences to shows where the average value came from, and to show which sentences to change to increase the average.

Finally, after each essay is done analysing. The last printouts will be the total number of sentences in the essay, the total sum value of the essay, and the average value of the essay. These printouts will summarized the value of the essay. If a student want to increase the value, then they can look through each paragraph to specifically find the sentence to change or to add more sentences.