Name: Ruida He

Student ID: 22762872

main function (text_file1, text_file2, feature)

according the feature you input to use the function below to get the result.

Read_file function (text_file)

1. Read the file and
2. make all the letter lowercase

Only_words function (text_file)

1. create a list starting from 0 to 127
2. delete the ASCII code of all the lowercase letters, " ' ", and "-"
3. use a loop to replace all the special characters that are not the lowercase letter, " ' ", and "-".
4. Use another loop to check which single quotes are needed to be saved (if the former character and the next character are lowercase letter, then save the single quote)
5. Split the string and return a list

Unigrams function (text_file)

1. Create a dictionary
2. Call the Only_words function, get the list of words
3. Use a loop and get function to append all the words into dictionary

Conjunction function (text_file)

1. Create a dictionary with all the conjunctions that has mentioned before
2. Call the Only_words function, get the list of words
3. Use a loop to check the word frequency of the conjunction in the dictionary.

Punctuation function (text_file)

1. Create a dictionary with all the punctuations that has mentioned before.
2. Use count function to check all the total number of “,”, “-“, “;”
3. For “’”, we just use the rules that we has applied in the Only_words function and count the number of single quote.

Composite function(text_file)

1. Join the dictionary of conjunction and the punctuation
2. Use Unigrams function to calculate the total number of words in the file.
3. For the amount of the sentences, we just consider all the situation, like “. \n”, “. \t”, “. “, “. \'”, “. \"”, and use count function to count all the number of sentences.
4. The amount of paragraphs is 1 plus the occurrence of “\n\n”
5. Append them into the dictionary and return them.

Distance function (text_file1, text_file2, feature)

1. For conjunction, punctuation and composite, the operation is similar
2. Use corresponding function to get the dictionary of text_file1 and text_file2, and use a loop and the formula
$$dis = \sqrt{\sum (profile1[i] - profile1[i])^2}$$
3. For Unigrams,
   first you need to use a loop to go through profile1, if there are words which also exists in profile2, use the formula to calculate and delete the words in profile2.
   If there is not, just consider profile2[i] is 0 and use the formula.
   And then use another loop to check the profile2 because there are still some words which haven’t appeared in profile1.
4. Return the distance.