Manik Singh Sethi
Software Design Section 2
Write up for Mini-Project 3 – Text mining and analysis

Overview – The original intent of this project was to analyze the similarities between texts professionally translated vs the output of web translators doing the same task. The selected text was Dante's Divine Comedy – Inferno and originally, I was going to use three different languages, Italian (original), English, and German. I was going to format the plain text so I could send individual sentences to Google translate and recompile into a string for a similarity test. The similarity function was found online and is documented as such. The intended result was to show how some languages are more accurate to translate to english without professional means.

Implementation – The basic overview of the process begins with downloading the texts from Project Gutenberg and pickling them so they may be accessed quickly and without multiple downloads. Then the texts are put through a couple functions to clean them of excess information from the plain text file such that only the content, arranged in cantos, are being used. Also, special characters and those with accents are removed and replaced with standard characters, respectively. Then, each text is split into its cantos, which are then split further into individual sentences because there is only a limited amount that the link can translate at once. After storing this information as a collection of lists, the italian sentences are then sent one by one to google translate and the results are then stored in list containing cantos which are represented as lists of sentences. At this point, the lists and strings are all combined to create two strings, one with the entirety of the official english text and the other with the entirety of the italian text that has been translated to english courtesy of google. Finally, the strings are compared to see how similar they are.

I had to reduce the amount of language as the formatting for proper translating was extremely time consuming so instead of comparing english with the translated italian and german texts, I simplified to only using english and italian texts. Another important decision I made was to store the translations on file and pull from the file as opposed to asking for translations from google each time. That process took nearly ten minutes each time and timeouts did occur occasionally. With them stored on file, the entire process was greatly sped up to taking less than 10 seconds for the whole program to run. In terms of comprehension however, I had to decide to run similarity tests on the entire text as opposed to each individual pair of sentences. This is because it seemed to be that the translations were spitting back seemingly random words as sentences at times and that may just be a product of google's comprehension of text. Any mismatch would throw off the entire program.

Results – the returning figure from the comparision of the two texts is 0.0059 out of 1. This shows an increadibly poor correlation between the two texts even though they are both different renditions of the same piece of work. For reference, the following examples were created on using the same function and method in terminal.

ex.
a = 'Just a small town girl living in a lonely world'
b = 'Just a small town girl living in a lonely world'
score = 1.0
an exact match

ex.
a = 'Just a small town girl living in a lonely world'
b = 'Just a big town boy living in a crowded world'

score = 0.7391
the match is clear but changes are noticed well

ex.
a = 'Just a small town girl living in a lonely world'
b = 'this hit that ice cold michelle pfeiffer that white gold'
score = 0.3106
the match is much much worse but still, there are connections being made

a = 'Just a small town girl living in a lonely world'
b = 'juste une fille dune petite ville vivant dans un monde tout seule'
score = 0.232
French and English sentence constructions and words are similar to show some sort of correlation

a = 'Just a small town girl living in a lonely world'
b = '01101010 01110101 01110011 01110100 00100000 01100001 00100000 01110011 01101101 01100001 01101100 01101100 00100000 01110100 01101111 01110111 01101110 00100000 01100111 01101001 01110010 01101100 00100000 01101100 01101001 01110110 01101001 01101110 01100111 00100000 01101001 01101110 00100000 01100001 00100000 01101100 01101111 01101110 01100101 01101100 01111001 00100000 01110111 01101111 01110010 01101100 01100100' (a in binary)
score = 0.0
no correlation is found at all

The correlation found is very faint in the texts and that can be attributed to synomyms used by translators, differences in modern language (used by google) and old usage (used by Dante and in the official translation) the point that comes across is that using online translators can be frighteningly inaccurate, incomplete, and just wrong. Using the tools on individual words can be better and usign command of the language is often much more accurate.

Reflection – it seemed that I bit off a little more than I could chew with this project because going through all the text to ensure proper texts were being taken and analyzed was much more intensive than I thought. I had to abandon running the same on other languages because of this reason. I think it would be better for me to sudocode and think of an outline for the code before starting to write it linearly. Additionally, I had to manually go through the cantos and give indicators to where each canto started. There should be an easier and more automated way to do this and I believe I did stumble upon a “find next” function that could have allowed me to find specified and consequent instances of the start of cantos. However, I found it much after I could have used it so I decided to leave it as is.