Mini Project 3: Text mining

Junhyun Nam (June)

Project overview

For this mini project, text mining, I used some storybooks from Project Gutenberg as data source. With this data, I analyzed the frequencies of words and tried to find some characteristics of each books by analyzing most commonly used words in the books. Also, I made some sentences by using Markov analysis. I hoped that I could make some real natural sentences by my algorithm, but it turned out to be not.

Implementation

The first file, word_frequency.py, prints the most frequent top 10 words with their frequencies. To distinguish characteristics between books, I eliminated the most common top 100 words in English. (Source: Wikipedia) The first thing I did to write this program was split the whole text to words. Since the split() method splits strings by space, I had to remove some punctuation marks such as quotes, dots, and question marks. After that, I made a histogram that counts the frequencies of each words using dictionary data structure. I already did it before in class, so it was easy. However, the keys in dictionary were words and the values in dictionary were frequencies, so I had to reverse the dictionary to sort words by frequency. I sorted the frequencies in decreasing order and eliminated some frequencies that corresponding words are in the list of most common top 100 words. Finally, this program prints the most frequent top 10 words with their frequencies after the elimination.
My second program, markov_text_synthesis.py, prints a random sentence composed of the random words in input text. To implement this program, I had to store the relation between two consecutive words. So I used dictionary to store this relation. I stored previous word as a key and the list of possible next words as a value, so that I can access the possible next words of specific words. In the process of convert text to words, I didn't remove some punctuation marks such as full stops, exclamation marks, and question marks that end sentences. That's because I had to distinguish when the sentence ends. So the sentences made by this program can end with only the words include such punctuation mark at the end. So after I split the text to words and made dictionary with them, I just generated the sentences. First, I randomly chose word that start with a capital letter. After that I iterated find next possible words using the dictionary and keep add to the first word. Finally, this process ends when the chosen word has full stop or exclamation mark or question mark.

Results

Following tables are the top 10 common words in each book.

Word	Frequency
at	4055
this	3971
man	1899
said	1791
marius	1352

jean	1176
more	1128
himself	1079
valjean	1046
then	1034

Table 1. Word frequencies in Les Miserables

Word	Frequency
said	1230
mr	1079
at	1005
this	847
oliver	769
very	491
upon	481
replied	464
old	448
man	365

Table 2. Word frequencies in Oliver Twist

Word	Frequency
tom	704
at	385
said	354
this	335
then	299
huck	217
don't	215
got	176
down	163
boys	154

Table 3. Word frequencies in The Adventures of Tom Sawyer

Word	Frequency
at	337
this	274
said	207
upon	203
man	132
holmes	95
project	88
more	85
then	84
little	83

Table 4. Word frequencies in A Study in Scarlet

Word	Frequency
alice	160
said	140
at	89
little	79
this	73
very	67
down	65
again	44
then	39
I'm	37

Table 5. Word frequencies in Alice's Adventures in Wonderland

After eliminating the top 100 common words, we can see some of left words are related to the characters. With this information, we can also guess who the speaker of this story is. For example, in A Study in Scarlet, Holmes's name appears quite many times, but Watson's is not. That may due to the speaker of the story is Watson.

The following sentences are generated from markov_text_synthesis.py

It was a breach in souls said, judging from the change of the day, at least, as he had no refuge. (from Les Miserables)
I could escape from his secretary, Stangerson, cried the ring. (from A Study in Scarlet)
I hardly hear the goose, with pink eyes half down it. (from Alice's Adventures in Wonderland)

Sometimes it generates natural sentences, but usually it generates awkward sentences that don't make sense.

Reflection

This was first time to me to deal with texts. I've heard about text mining before, but I've never tried to do it. So first, thanks to you for choosing this topic for the mini project 3. During this project, the most difficult thing was the way to split text into words or sentences. To split text to words was easier than to sentences since I didn't have to conserve punctuation marks. However, to split text to words was so complicated since there were too many cases that I had to consider. The most important thing was to figure out when the sentence ends. But there were so many kinds of marks and each marks had different usage. That made this project harder. For example, '.' can used as full stop, but also used in acronyms. If I can use better algorithm to split text into sentences, I can wrote better program for Markov text synthesis.
Also there were so many useless strings to make sentences in the text, such as *** or some numbers represented in Roman, and names of chapters written in capital. In my program, the sentence starts with capital letter. So there was some problem to choose proper word at first due to those useless strings.

But after convert text to words or sentences, rest of the process were not difficult. I just stored them or their relation in the dictionary and used them. So I think the most important and difficult stuff in text mining and analysis is converting those texts of irregular format to some neat format. If I could use better algorithm to do that, my result would be much better.