Edwin Hu

Ling165

Professor Koo

**Ling165 Final Project**

**Introduction**

This project is trying to build an application that can evaluate the difficulty levels of an article. It is important for students who are learning English to find articles that fit their level so that will not spend too much time on the articles that are too hard or too easy for them. It will also be useful for English teachers to use this application to design a lesson plan for reading class, and choose an article that is suitable for the class.

**Data Set**

Data set used in this project is from New York Times. I got all the articles from April 2016 to April 2017 using the API provided by New York Times and counted the frequency of each word. In order to make the frequency list concise and efficient, I removed all the stopword. The list of stopword is in file stopword.txt.

I read each article line by line and split each line by space, then remove all the punctuations and converted all the letters into lower case. The result data set contains around 50,000 words with the most frequent word “president” which appears 4335 times. Please see the file wordfreq.txt for the complete list.

|  |  |
| --- | --- |
| Word | Frequency |
| people | 2267 |
| today | 413 |
| know | 1431 |
| … | … |

Table 1. Word frequency table

**Implementation**

So how am I going to use the wordfreq list? I am going to write a code to make the article I going to evaluate into a list, then compare the word in the target article with the wordfreq list and count the difficulty score. The formula I am going to use count the score is the following:

* WC = Word Count, WS = Word Score
* WC1 \* WS1 + WC2 \* WS2 + WC3 \* WS3 + ……
* (WC1  + WC2  + WC3 + ……)
* = Difficulty Score

So the function getScore in main.py is doing the above formula. I count the word in the article and compare its frequency in the wordfreq list which is the word score. Then I multiply the two numbers and divided by the sum of word that have used in the article. If the difficulty score is high, then it means that the article is comparably easy. The lower the difficulty score is, the harder the article will be. That is the ideal speculation by my code. The function getWordFreq in the main.py is simply put the wordfreq list into dictionary so the getScore function can use it to compare.

**Evaluation**

After implementing the code, the most challenging thing for this project is to determine the difficulty level. How to interpret the difficulty score to see if the article is easy or hard? I tested many articles from New York Times and some articles from the English teaching websites where the articles were already classified with the difficulty levels. The highest score I got is around 500, and the lowest is around 300. So I chopped the scores from 300 to 500 into 10 blocks and gave it a difficulty level instead of the difficulty score.

However, there are many challenges I faced when doing the evaluation. First, I found that I need more data, and more data from different sources. The reason to that is when I look into the wordfreq list, I found that there are some words that are usually considered as easy words but were tagged with low frequency. This will hurts the precision of the difficulty score because easy words did not get a high score. Some basic vocabulary did not get used too much by the N.Y times. In order to keep my statement from the begging that “higher frequency means the word is easier” reliable, I need to collect more data and from more resources so the dataset can be more complete.

Another thing I need to consider is stemming which means words with different suffix or type but have the same meaning should count as the same word. This will make the wordfreq list more reliable. All the above problems had been mentioned when I was presenting my project. However, it is not easy to implement it when I am actually writing the code and collecting the data. I will keep the above challenges in mind and improve this project in the future.