Report - Plagiarism Checker

Utkarsh Munjal - 2018CS10395

1 How to Run The Code

unzip 2018CS10395.zip to get 2018CS10395_plagchecker
To compile and link run make command, it will create executable named **plagChecker** the same directory
To run the executable :

```
./plagChecker<SPACE><LOCATION_OF_TEST_FILE><SPACE><LOCATION_OF_CORPUS_FOLDER>
```

This will give the output on console in the format

```
<TEXT_DOCUMENT_NAME> <SPACE> <SIMILARITY_PERCENTAGE>
```

Note: only the files ending with **.txt** are read.

2 Implementation

2.1 Gram Formation and Storing

The bigrams and trigrams (word level) are formed for the file to be tested. Text file is parsed and whenever a space is found a word is completed, now this completed word is combined with the previous word to form a bigram and that bigram is combined with the word previous to that to form the trigram. Now these bi-grams and tri-grams are stored in a linkedList called mainGram, if the gram is already present in the linkedList then the frequency of that is increased otherwise a new node is created. While inserting a gram in linkedList the sorting of list is maintained in accordance with the comparator defined in the code (compareStrings()).
The worst case time complexity for inserting one gram is $O(n)$ where n is the number of grams. So for storing all the grams worst case complexity is $O(n^2)$. If number of words in a document are D number of k-grams formed are $D - K + 1$. Hence for k = 2,3: $n = \theta(D)$. Hence the time taken to store all the grams is $O(D^2)$. So the combined time complexity for parsing, gram formation, and storing all grams is $O(D^3)$.
Following the similiar approach, gram formation takes place for corpus files one after another. Once similarity for one corpus file is calculated after that the gram formation of next starts. The space complexity is equal to $O(n_{max})$, where $n_{max} = max(n_{test}, n_1, n_2, n_3, ...,)$ where n_i is the number of grams formed by i^{th} file. The worst case space complexity in terms of number of words is $O(D_{max})$ where $D_{max} = max(D_{test}, D_1, D_2, D_3, ...,)$ where D_i is the number of words in the i^{th} file. . Note that the worst case for space complexity is the case in which all the bi-grams and tri-grams formed are distinct. For same bi-gram or tri-gram space is only allocated only once.

2.2 Calculating Similarity

Once the grams for the test file and corpus file are ready. Both the linkedLists are traversed, intersection and union of grams are found for both the files. Since the lists are already sorted it is to find the intersection and union (same as merging in merge sort). For any term t, union[t] = wt(t)*max(a[t],b[t]) and intersection[t] = wt(t)*min(a[t],b[t]), where a[t] and b[t] are number of times gram t occurred in files a and b resp.; wt(t) is weight assigned to term t. To compensate for the frequent occurrence of small strings weight is assigned in proportion on to string length.

$$wt(t) = log\left(1 + \frac{strlen(t)}{9.0}\right)$$

$$intersection[t] = wt(t) \cdot min(a[t], b[t])$$

$$union[t] = wt(t) \cdot max(a[t], b[t])$$

Note that average length of a word in english ≈ 4.7. And here we are considering bi-grams (two words joined) and tri-grams (three words joined).

$$\text{Similarity} = \left(\frac{\sum \text{intersection[t]}}{\sum \text{union[t]}}\right) * 100\%$$

The similarity used here is similar to Jaccard Index, given by :

$$J = \frac{|A \cap B|}{|A \cup B|}$$

The time complexity for calculating the similarity is $O(max(n_a, n_b))$ where n_i is number of grams in i^{th} file.