CS1003 W05 Practical 130011035 Tutor: Dharini 27 Feb 2014

Introduction:
This weeks practical asked us to design a system which would: read in a file; dissect individual words; count the number of times each of the words occurred; and output to another file the word, number of occurrences, rank (how this words number of occurrences compares to the rest of the words), probability and the zipf(the probability of a word multiplied by the rank). When these statistics were all combined, they generated the zipf which is the main reason for the practical, to test if the zipf theory holds up. The zipf law states that the probability of a word occurring at a given location in a text document is inversely proportional to the words overall rank in the document.

$$\text{Rank} \times \text{Probability} = \text{constant}$$

In general for English documents, constant ≈ 0.1.

Extension tasks were attempted with graphs being produced for the 3 csv documents which were provided. These are all named accordingly and are stored in the 'Output' folder.

Testing:

| Input | Expected Output | Success |
|---|---|---|
| Processing the 'Duel on Syrtis' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'Hound of the Baskervilles' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'Mysterious Affair at Styles' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'The Duelling Machine' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'The Repairman' document | A csv file is produced, with all the correctly | Yes |

| | | |
|---|---|---|
| | displayed output | |
| Processing the 'Works of Edgar Allan Poe' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'A Midsummer Night's Dream' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'Alls Well That Ends Well' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'Antony & Cleopatra' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'As You Like It' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'Coriolanus' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'Hamlet' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'One Day More' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'Works of Edgar Allan Poe' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'The Comedy of Errors' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'The Merchant of Venice' document | A csv file is produced, with all the correctly displayed output | Yes |
| Processing the 'Twelfth Night' document | A csv file is produced, with all the correctly displayed output | Yes |

Problems encountered:

During the coding stage I initially attempted to use an array list to store the words, as I believed that the addition of new words would be easier this way. This quickly fell apart when it came to sorting and searching of the array, and so the array list was exchanged for a normal array.

Both methods were used: SortWords, and clean word

Also during development, I experienced problems getting loops to work, as null pointers began to be experienced, but a few if statements sorted those.

Extensions attempted were graphs which are held in output along with the csv files.