5/29/2019

Document Statistics
Design Doc

Designer Name: Dawson Chen Class: AP CS w/ DS P7

Description

The DocumentStats class offers services to determine the author of a given document. A list of potential authors, 5 different statistics of theirs, and the corresponding statistics of the given document must be provided and taken into account.

Internal Data Structures

```
private Document document;
private double[] weights = {0, 11, 33, 50, 0.4, 4};
private double[] docStats;
private Map<String, List<Double>> authors;
```

- `Document document` represents the document that is being evaluated and whose statistics are being generated.
- `double[] weights` contains the predetermined weights, or importance, of each of the statistics of the document as well as the authors.
- `double[] docStats` contains all the statistics(listed below) of the document, which are computed in `computeStats()`.
    a. Author's name
    b. The documents' average word length
    c. Type-Token Ratio
    d. Hapax Legomena Ratio
    e. The average sentence length
    f. The average number of phrases per sentence
- `Map<String, List<Double>> authors` is of Map of keys representing names of authors and values representing a list of their statistics

Constructor

```
public DocumentStats(Document doc, Map<String, File> f) throws
IOException
```

- The DocumentStats constructor takes in 2 parameters. It sets all the instance variables of the class to their appropriate values and then loads all the statistics of the authors into the authors map.
- Parameters
    - `Document doc` is the document to be evaluated and analyzed.
    - `Map<String, File> f` is a Map of keys representing file names and values representing the File object that the file name corresponds to.

- Functionalities and Notes
    - It sets `document` to be the `doc` parameter.
    - It sets docStats[] to a new array of doubles of length 6. It will then be populated with the 6 statistics of the document in `computeStats()`.
    - It sets the `authors` map to the return value of `loadAuthorStats(f)`, which loads the statistics of each author into `authors` after fetching them from each of the files in `f`, the Map parameter.
    - Further details about `loadAuthorStats(Map<String, File> files)` will be explained below.
    - It throws an IOException since `loadAuthorStats(Map<String, File> files)` is called in the constructor and also throws an IOException

Methods

```
public Map<String, List<Double>> loadAuthorStats(
     Map<String, File> files) throws IOException
```

- This method retrieves each author's statistics from the `files` parameter and returns a map holding each author's name and statistics, which is then loaded into `authors` in the constructor.

- Parameters
    - `Map<String, File> files` is a Map of keys representing file names and values representing the File object that the file name corresponds to.

- Functionalities, the Return Value, and Notes
    - It loops through the `files` parameter, reads each file name, and extracts each statistic of each file, and puts it into a `Map<String, List<Double>>` which is returned after files is entirely looped through.
    - It throws an IOException because it uses a FileReader to read `files` and causes the exception whenever the input is failed or interpreted.
    - Runs in constant time or O(n) depending on the number of authors needed to load

`public void computeStats()`

- This method calculates each statistic of `document` by looping through its sentences, phrases, and tokens. It determines each statistic in the following ways.
    a. *The Author's name* is neglected because it has a weight of 0, and therefore has no influence
    b. *The documents' average word length* is calculated by summing all the lengths of all the words and dividing by the total number of words.
    c. *Type-Token Ratio* is calculated by dividing the number of different words by the total number of words in the document. The number of different words is calculated by adding each word to a set, and since sets take no duplicates, the set's size after iterating through the different words is the number of different words.
    d. *Hapax Legomena Ratio* the number of words occurring exactly once in the text divided by the number of words in the document. The number of words occurring exactly once in the text is found by looping through all the words and add each first appearance of a word to a list and removing the word if a second appearance is seen.
    e. *The average sentence length* is found by summing the total number of words and dividing by the total number of sentences.
    f. *The average number of phrases per sentence* is found by summing up the total number of phrases and dividing by the total number of sentences.
- The statistics are then added to the docStats array.
- Runs in O(n) where n is the number of tokens in the document

`public String findAuthor()`

- This method determines the most likely author of `document` out of all the possible authors listed in `authors.`

- Return value, Functionality, and Notes
    - This method returns the name of the most likely author of `document`

- It loops through the `authors` and determines which author's statistics are most different from `document`'s. Each of the statistics' differences are multiplied by their corresponding weight to account for which statistics are more important.
- Runs in O(n) depending on the number of authors

Other Notes

DocumentStatistics can be used by the ScannerTester class to determine the most likely author of a document provided during this class's construction. The ScannerTester's main method passes in a map of file names and File objects, obtained using the listFiles() method on the directory "./SignatureFiles/" and adding the contents of listFiles() to the map through a for loop.