Document Statistics Design Document

Shounak Ghosh, Period 7

Description

The Document Statistics class parses a given document (passed in through the constructor), computes the average word length, the type-token ratio, the Hapax-Legomana ratio, the average number of words per sentence, and the sentence complexity of the given piece of text. After these statistics have been computed, they are compared to a collection of signature statistics from known authors, and a prediction of the author of the document is drawn based on comparisons between the signature statistics and the document statistics.

Services

- `double getAverageWordLength()`
  - Divides the total length of all the words by the total number of words
  - Has constant runtime; both values mentioned above are computed upon parsing the document
- `double getTypeTokenRatio()`
  - Computes the ratio between the number of different words in the document to the total number of words in the document
  - Has constant runtime; the number of different words is also computed upon parsing the document
- `double getHapaxLegomanaRatio()`
  - Computes the ratio between the number of unique words in the document to the total number of words in the document
  - Has linear runtime; the number of unique words is computed using a set of all the words in the document, along with a list of all the words in the document
- `double getAverageWordsPerSentence()`
  - Divides the total number of words by the total number of sentences
  - Has constant runtime; the total number of sentences is computed upon parsing the document

- `double getAverageSentenceComplexity()`
  - Divides the total number of phrases by the total number of sentences
  - Has constant runtime; the total number of phrases is also computed upon parsing the document
- `double[] getDocumentStatistics()`
  - Retrieves a double array of all the computed statistics
  - Has constant runtime; A shallow copy is returned
- `String findAuthor()`
  - Compares the document statistics to the signature author statistics, and returns the best match from the given authors
  - Has a linear runtime; The number of comparisons is dependent on the number of known author statistics

Internal Data Structures and State

- `Document document`
  - The document to predict the author of (passed in through the constructor)
- `ArrayList<AuthorStats> sigStats`
  - An ArrayList that stores information about the given author statistics
- `ArrayList<Token> words`
  - An ArrayList of all the words in the document
- `HashSet<Token> wordSet`
  - A HashSet of all the different words in the document
- `double[] weights`
  - The multiplicative weights applied to the statistics when predicting the author of the document
- `double[] docStats`
  - A double array of all the statistics computed from the document
- `BufferedReader f`
  - The file reader used to read in the signature author statistics
- `Scanner sc`
  - A `java.util.Scanner` object used to parse the information read in from the `BufferedReader f`
- `String name`
  - String variable used to store the name of the author while reading from the given signature statistics files
- `double avgWordLen`

    - Double variable used to store the average word length while reading from the given signature statistics files
- ➢ `double typeTokenRatio`
    - Double variable used to store the type-token ratio while reading from the given signature statistics files
- ➢ `double hapLegoRatio`
    - Double variable used to store the Hapax-Legomana ratio while reading from the given signature statistics files
- ➢ `double avgSentenceLen`
    - Double variable used to store the average sentence length while reading from the given signature statistics files
- ➢ `double avgSentenceComplexity`
    - Double variable used to store the average sentence complexity while reading from the given signature statistics files

Test Plan

The DocumentStatistics class can be tested by predicting the authors of multiple "mystery" (the correct authors are known to the testers) texts by comparing the calculated statistics to the collection of signature statistics from various authors. The predicted author is the closest match between the document statistics and the signature statistics. The class is considered to be correct if it is able to correctly identify the authors of at least five different mystery texts.