calculating TF-IDF

Codes

Tokenizer

the tokenizer is descriped in last project

TF(documents,outputFormat='sparseMatrix')

- **Parameters:** documents: string, list an string contains a folder name of corpus or list of documents outputFormat: string 'sparseMatrix' or 'pandas_DataFrame'
- **Returns:**
  an sparse matrix or `pandas.DataFrame` object of term frequency calculated.

In order to calculate the TF of the documents, several steps are taken as descriped below: - Fistly, each file will be opened and read using `open` module. - Secondly, the data extracted from files are passed to `tokenizer.wordTokenizer` which is implemented in the previous assignment.this step will convert the text into the list of tokens. - In the third step,the list of tokens are passed to `tokenizer.wordCounter` in order to count the frequency of each token in the current document. This finction is also implemented in the previous assignment. The output will be a `dict` object which maps tokens into their frequency. - As the next step, The last calculated dictionary is stored as the value of another dictionary. This nested dictionary structures is forming a sparse matrix which saves more space than regular matrix. - Finally, the function will return the output matrix as `pandas.DataFrame` if needed.

DF(documents,outputFormat='sparseMatrix')

- **Parameters:** documents: string, list an string contains a folder name of corpus or list of documents outputFormat: string 'sparseMatrix' or 'pandas_DataFrame'

- **Returns:**
  map of tokens to their DocumentFrequency of `pandas.DataFrame`

In order to calculate the DF of the documents, several steps are taken as descriped below: - Fistly, each file will be opened and read using `open` module. - Secondly, the data extracted from files are passed to `tokenizer.wordTokenizer` which is implemented in the previous assignment.this step will convert the text into the list of tokens. - In the third step, the frequency of each token will increase by one value if that token is in the current document. - Finally, the function will return the output matrix as `pandas.DataFrame` if needed.

TF_IDF(TF_mat,DF_mat,outputFormat='sparseMatrix')

- Parameters:

  ```
   TF_mat:
  the output of TF function
  DF_mat:
  the output of DF or DF_fromTF function
  outputFormat: string
  'sparseMatrix' or 'pandas_DataFrame'
  ```

- Returns:
  an sparse matrix or `pandas.DataFrame` object of TF-IDF calculated.

Having TF and DF will makes calculation of the TF-IDF pretty easy. simply each row of the TF matrix is divided by the corresponding term in the DF matrix.

calculate DF matrix using TF matrix

ther is another faster way to calculate DF matrix and that is using TF matrix.To do so,number of non-zero values will be count in each line of the TF matrix. This operation is incredibly fast using `pandas.DataFrame`: - First: divide the TF matrix by itself - Second: add numbers in each line and save the answer as the DocumentFrequency of terms.

Test

There is two test corpus available: - CorpusSmall: contains two small files types by my self just to test the outputs. - CorpusBig: contains 150 file which each files has 16 lines.

Output

there is brief information in standard output containing execution time and small view of the matrixes.

Execution Time:

```
 The TF matrix is calculated in 4.823282718658447

The DF matrix is calculated in 0.007876873016357422

The TF-IDF matrix is calculated in 0.15381646156311035
```

obviously calculating the TF requires the hard drive file access, which makes it slowly.

Saved Files

There is three saved files `tf.xlsx`, `df.xlsx`, `tf_idf.xlsx`.(their names are clear enough to decribe their contents)

tf.xlsx

5045 rows x 150 columns rows contains tokens. columns contains documents.

DF.xlsx

5046 tokens are found and sorted.

tf_idf.xlsx

5045 rows x 150 columns rows contains tokens. columns contains documents.