

ReadMe_Word Frequency Analysis

Subject:

Data Science Assignment for
M.Sc. Information Studies - Data Science

Author:

Rodion Burden

Technology used:

- Html: standard markup language
 - JavaScript (ES6+): ECMAScript based scripting language with prototype-based object-orientation.
 - PlotLy: Modern Analytics library. Suitable for JS and Python.
-

Code Description

(app.js, Appendix 1)

The code is designed as an html parser that extracts (html-)content from a webpage and processes it according to the required specifications.

The created class 'TextAnalyzer' consists of three parts, based on the role within the application.

1. Handle Data
 - Get data from the document object model (DOM)
 - Client the text
 - Store the texts
 - Calculate word frequencies
 2. Handle Wordquery
 - Check presence of the word
 - Create hash table as response to the query
 3. Create Histogram
 - Prepare data
 - Create histogram with PlotLy
-

Output Samples

ADD 2: HANDLE WORDQUERY

Instantiate the class

```
const analysis = new TextAnalyzer();
```

Query the word 'the'

```
analysis.analyze('the');
```

Result in the browser:

Analysis

The word "the" occurs in article 1: 94 times,
in article 2: 12 times and
in article 3: 65 times. In total it occurs 171.

ADD 3: CREATE HISTOGRAM

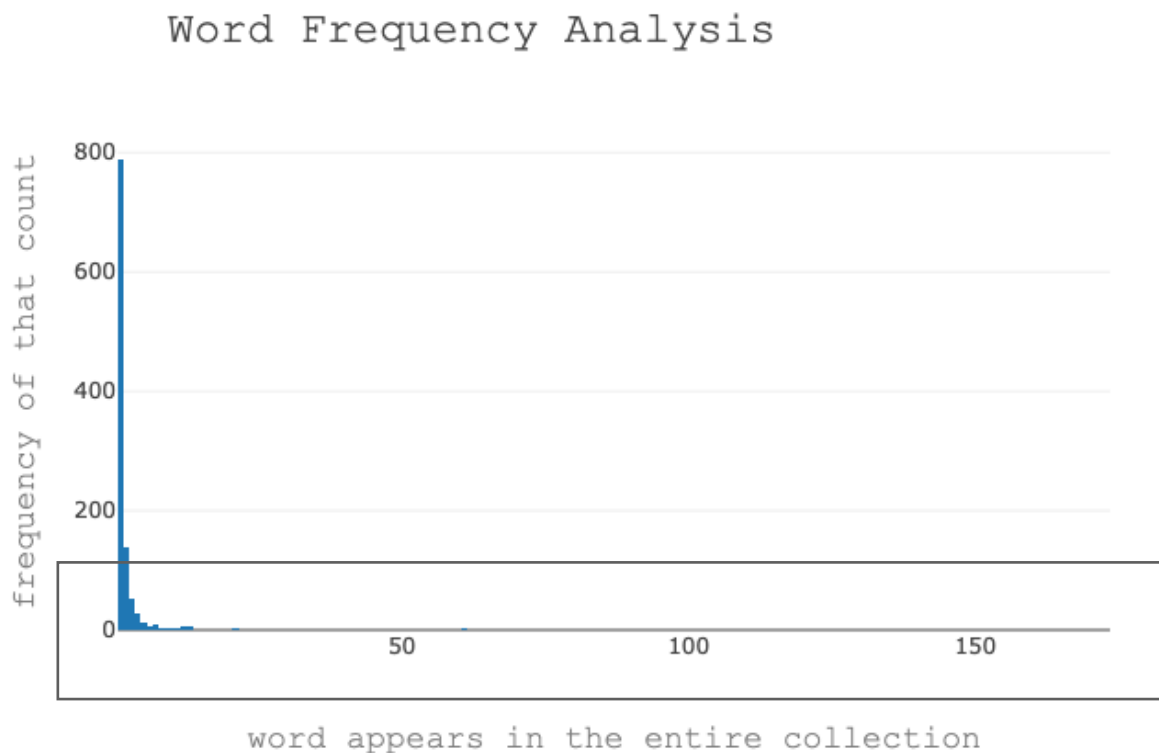
Prepare data

```
analysis.allTexts(DOM.plainText);
```

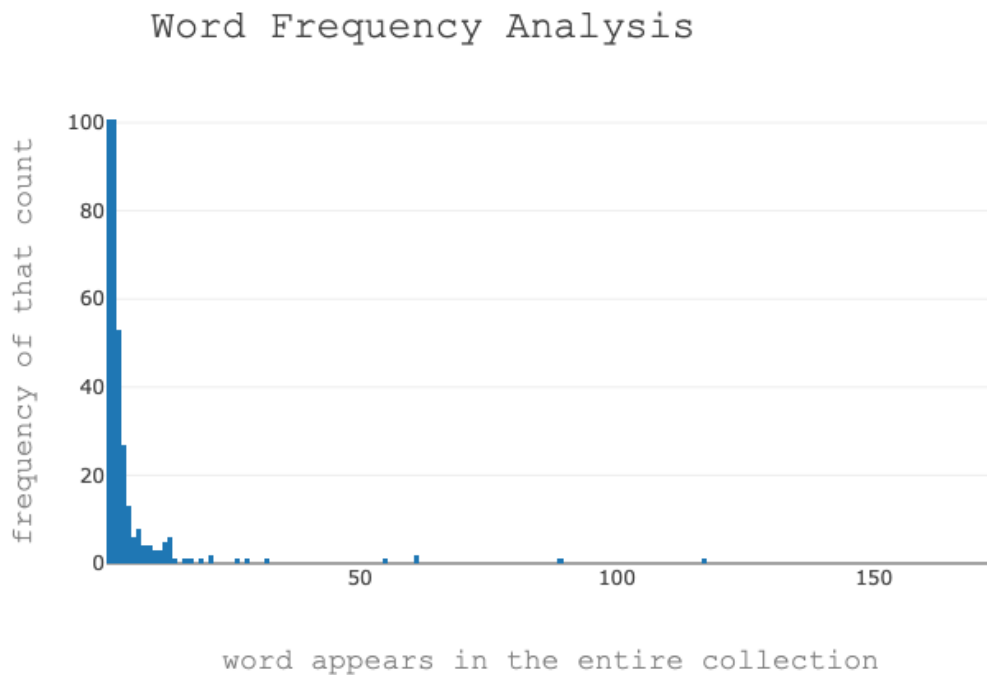
Call the method to create the histogram

```
analysis.renderHistogram(DOM.histogram);
```

Result in the browser:



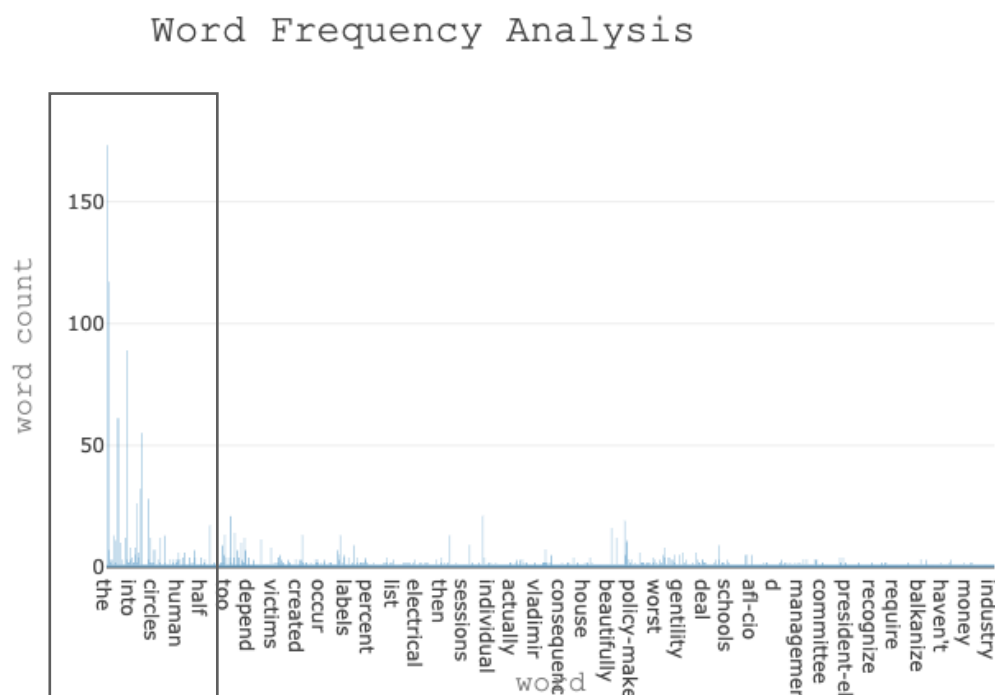
Zoom

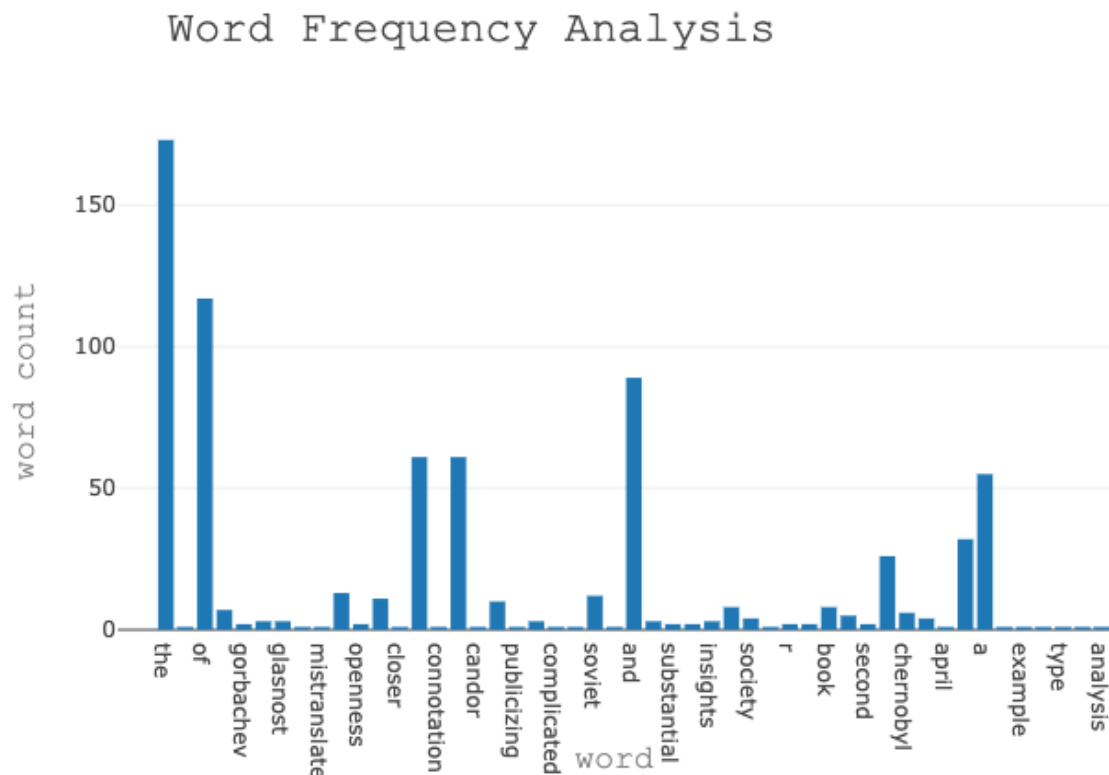


Interpretation

“The histogram illustrates well Zipf’s law, stating that the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc.” The second histogram provides a deeper look and illustrates further implications of Zipf’s law.

```
analysis.renderHistogram(DOM.histogram2);
```





"In the Brown Corpus of American English text, the word the is the most frequently occurring word, and by itself accounts for nearly 7% of all word occurrences (69,971 out of slightly over 1 million). True to Zipf's Law, the second-place word of accounts for slightly over 3.5% of words (36,411 occurrences), followed by and (28,852). Only 135 vocabulary items are needed to account for half the Brown Corpus."

The analysis of the given texts are in line with the overall observed frequency of the three most common word 'the, of, and'.

(Source: https://en.wikipedia.org/wiki/Zipf%27s_law)

Appendix 1

```

1 DOM = {
2   analysis: document.querySelector('.output1'),
3   plainText: document.querySelector('.output2'),
4   histogram: document.querySelector('.output3'),
5   histogram2: document.querySelector('.output4')
6
7 };
8
9 class TextAnalyzer {
10
11   texts = {};
12   wordFreqsPerArticle = {};
13   wordQueryOutput = [];
14
15   allTextsInOne = {};
16   wordFreqsAllArticles = {};
17
18   // ***** Handle Data
19   //get articles from DOM
20   getArticles = () => {
21     this.articles = [...document.getElementsByTagName("
22 text")];
23   };
24
25   // clean text in article
26   cleanText = (s) => {
27     // Normalize
28     s = s.toLowerCase();
29     // Strip quotes and brackets
30     s = s.replace(/["'"(\{\}\})|\B['"]([\^'"]+)[']]/g,
31 '$1');
32     // Strip dashes and ellipses
33     s = s.replace(/[\----...]|--|\.\.\./g, ' ');
34     // Strip punctuation marks
35     s = s.replace(/[!?:;.,]\B/g, '');
36     // remove html tags [<p>, </p>]
37     s = s.replace(<p>/g, '');
38     s = s.replace(<\p>/g, '');
39     return s;
40   };
41
42   getWordsfromArticle = (s, i) => {
43     s = this.cleanText(s);
44     this.texts[i] = s;
45   };
46
47   // store texts of every article
48   storeWordsfromArticles = (articlesObject) => {
49     Object.keys(articlesObject).forEach(articleKey => {
50       this.getWordsfromArticle(articlesObject[
51 articleKey].innerHTML, parseInt(articleKey)+1)
52     });
53   };
54 }

```

```

53 // calc word frequency in individual article text
54 calcWordFreq = (s, i, storeFreqs) => {
55   storeFreqs[i] = s.match(/\S+/g).reduce(function(
oFreq, sWord) {
56     if (oFreq.hasOwnProperty(sWord)) ++oFreq[sWord];
57     else oFreq[sWord] = 1;
58     return oFreq;
59   }, {});
60 };
61
62 //store word frequency of every article
63 storeFreqsfromTexts = (textsObject, storeFreqs) => {
64   Object.keys(textsObject).forEach(textKey => {
65     this.calcWordFreq(textsObject[textKey],
textKey, storeFreqs);
66   })
67 };
68
69 // ***** Handle word query. Example: [the] -> [1, 20
] -> [2, 34] -> [3, 12]
70 // check for presence of a word in an object
71 isWordInSet = (word, set) => {
72   return (word in set);
73 };
74
75 // create output arrays showing location and frequency
of the word
76 showIfWordInSet = (word, article, wordset) => {
77   const isIncl = this.isWordInSet(word, article);
78   if(isIncl){
79     let arr = [wordset, article[`${word}`]];
80     this.wordQueryOutput.push(arr);
81   }
82 };
83
84 // create hashtable as response to word query
85 showWordInSets = (word, sets) => {
86   this.wordQueryOutput.push(word);
87   Object.keys(sets).forEach(wordSet =>{
88     this.showIfWordInSet(word, sets[wordSet],
wordSet);
89   })
90 };
91
92 // render the result of the word query to the DOM
93 renderOutputToDom = (output) => {
94   const element =
95     `<p>
96     The word "${output[0]}" occurs in article ${output
[1][0]}: ${output[1][1]} times, </br>
97     in article ${output[2][0]}: ${output[2][1]} times
and </br>
98     in article ${output[3][0]}: ${output[3][1]} times
99     .
    In total it occurs ${output[1][1]+output[2][1]+

```

```

99 output[3][1]}.
100     </p>`
101     DOM.analysis.innerHTML = element;
102 };
103
104     // In an improved version of the code the following
105     method would be redesigned
106     // to a computationally more efficient method, so the
107     data handling of
108     // all the articles doesnt run for every word query.
109     ie I would build in
110     // the parameters (constructor) into the class and run
111     the data handling
112     // at the instantiation of the class.
113     analyze = (word) => {
114         this.getArticles();
115         this.storeWordsfromArticles(this.articles);
116         this.storeFreqsfromTexts(this.texts, this.
117 wordFreqsPerArticle);
118         this.showWordInSets(word, this.wordFreqsPerArticle
119 );
120         this.renderOutputToDom(this.wordQueryOutput);
121     };
122
123     // ***** create histogram
124     // plain texts
125     allTexts = (out) => {
126         const keys = Object.keys(this.articles);
127         keys.forEach(key => {
128             out.appendChild(this.articles[key]);
129         });
130         out.innerHTML = this.cleanText(out.innerHTML);
131         this.allTextsInOne = this.cleanText(out.innerHTML)
132 ;
133         //remove remaining html tags and numbers
134         this.allTextsInOne = this.allTextsInOne.replace(/<
135 text>/g, '');
136         this.allTextsInOne = this.allTextsInOne.replace(
137 /<\/text>/g, '');
138         this.allTextsInOne = this.allTextsInOne.replace(/[
139 0-9]/g, '');
140         this.calcAllFreq(this.allTextsInOne);
141
142     };
143
144     calcAllFreq = (s) => {
145         this.wordFreqsAllArticles = s.match(/\S+/g).reduce
146 (function(oFreq, sWord) {
147             if (oFreq.hasOwnProperty(sWord)) ++oFreq
148 [sWord];
149             else oFreq[sWord] = 1;
150             return oFreq;
151         }, {});
152     };

```

```

142     renderHistogram = (DOMhistogram) => {
143         // transform object to array
144         const result1 = Object.keys(this.
wordFreqsAllArticles).map((key) => {
145             return [this.wordFreqsAllArticles[key]];
146         });
147
148
149         // create histogram
150         const arrRes = result1.map(el => {
151             return el[0];
152         });
153         const x = arrRes;
154         const trace = {
155             x: x,
156             type: 'histogram',
157
158         };
159         const data = [trace];
160         const layout = {
161             title: {
162                 text: 'Word Frequency Analysis',
163                 font: {
164                     family: 'Courier New, monospace',
165                     size: 24
166                 },
167                 xref: 'paper',
168                 x: 0.05,
169             },
170             xaxis: {
171                 title: {
172                     text: 'word appears in the entire
collection',
173                     font: {
174                         family: 'Courier New, monospace',
175                         size: 18,
176                         color: '#7f7f7f'
177                     }
178                 },
179             },
180             yaxis: {
181                 title: {
182                     text: 'frequency of that count',
183                     font: {
184                         family: 'Courier New, monospace',
185                         size: 18,
186                         color: '#7f7f7f'
187                     }
188                 }
189             }
190         };
191
192         Plotly.newPlot(DOMhistogram, data, layout);
193         /* Second histogram
194         const wordArr = this.allTextsInOne.match(/\S+/g);

```



```

195     const x2 = wordArr;
196     // -- this shows the actual words on the X axis of
the histogram
197     const trace2 = {
198         x: x2,
199         type: 'histogram',
200
201     };
202     const data2 = [trace2];
203     const layout2 = {
204         title: {
205             text: 'Word Frequency Analysis',
206             font: {
207                 family: 'Courier New, monospace',
208                 size: 24
209             },
210             xref: 'paper',
211             x: 0.05,
212         },
213         xaxis: {
214             title: {
215                 text: 'word',
216                 font: {
217                     family: 'Courier New,
monospace',
218                     size: 18,
219                     color: '#7f7f7f'
220                 }
221             },
222         },
223         yaxis: {
224             title: {
225                 text: 'word count',
226                 font: {
227                     family: 'Courier New,
monospace',
228                     size: 18,
229                     color: '#7f7f7f'
230                 }
231             }
232         }
233     };
234     Plotly.newPlot(DOM.histogram, data2, layout2);
235
236 }
237
238 }
239
240 const analysis = new TextAnalyzer();
241 analysis.analyze('the');
242 analysis.allTexts(DOM.plainText);
243 analysis.renderHistogram(DOM.histogram);
244 analysis.renderHistogram(DOM.histogram2);
245
246

```