

---

**IT-309**  
**EXPERIMENT 1 AND 2 REPORT**

---

**Indian Institute of Information Technology, Vadodara**

**Group-1**

# Contents

0.1	Experiment -1 - WORD CLOUD . . . . .	2
0.1.1	First fifty most frequently appearing words . . . . .	2
0.1.2	Histogram of first fifty most frequently appearing words . . . . .	3
0.1.3	Validating Zipf's Law . . . . .	3
0.2	Experiment -2 - Building Term-Document Matrix (TF-IDF) . . . . .	4

## 0.1 EXPERIMENT -1 - WORD CLOUD

- In this experiment, we build a Hadoop cluster of 7 nodes.
- We took our corpus as **All books of Harry Potter** *collectivesize > 7MB*
- We implemented MapReduce in Java for word count.
- For plotting graphs and curve for Zipf's law, we used Matlab.

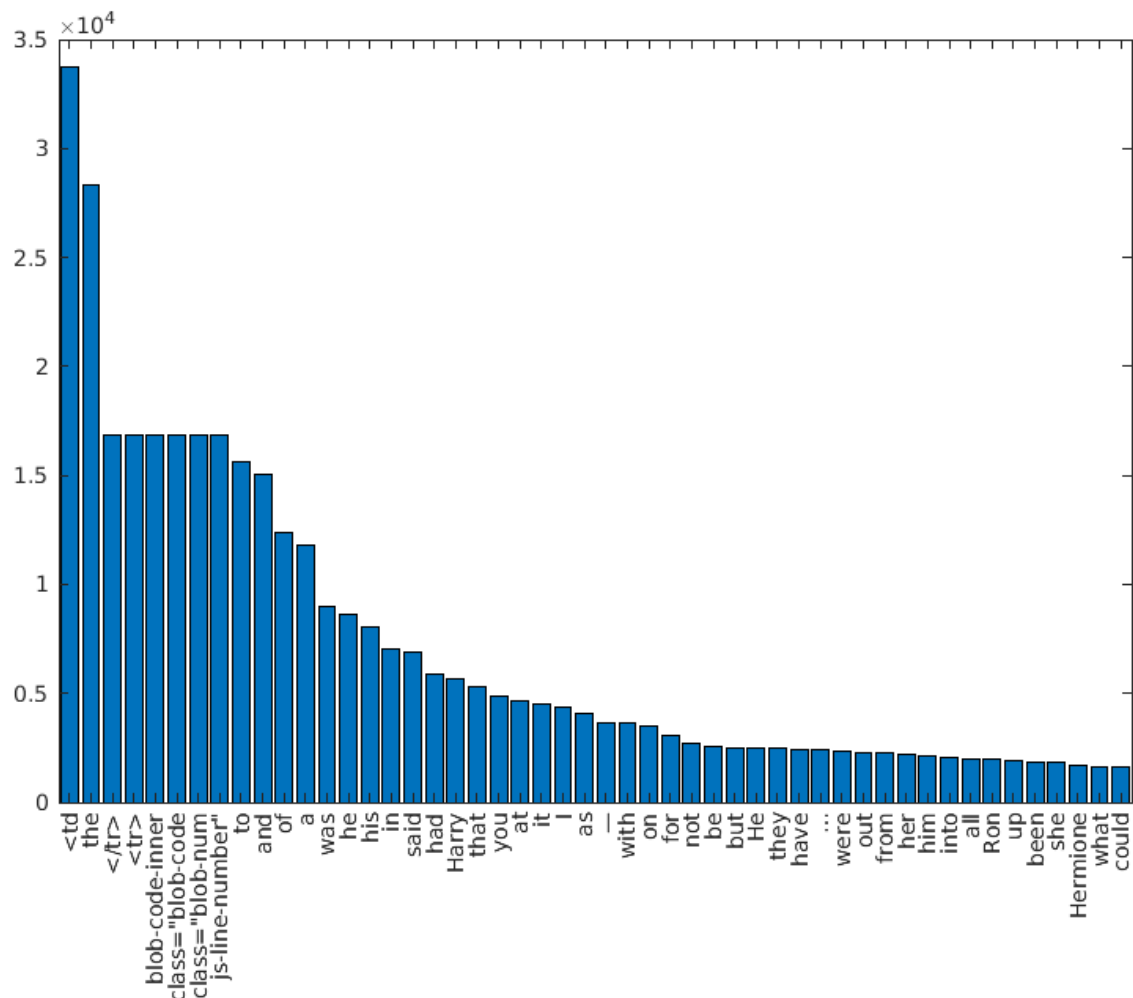
### 0.1.1 First fifty most frequently appearing words

Some of the most common words are shown below:

'<td'	33750
'the'	28294
'</tr>'	16875
'<tr>'	16875
'blob-code-inner'	16875
'class="blob-code'	16875
'class="blob-num'	16875
'js-line-number"'	16875
'to'	15629
'and'	15011
'of'	12375
'a'	11834
'was'	8976
'he'	8619
'his'	8053
'in'	7005
'said'	6917
'had'	5857
'Harry'	5657
'that'	5321

### 0.1.2 Histogram of first fifty most frequently appearing words

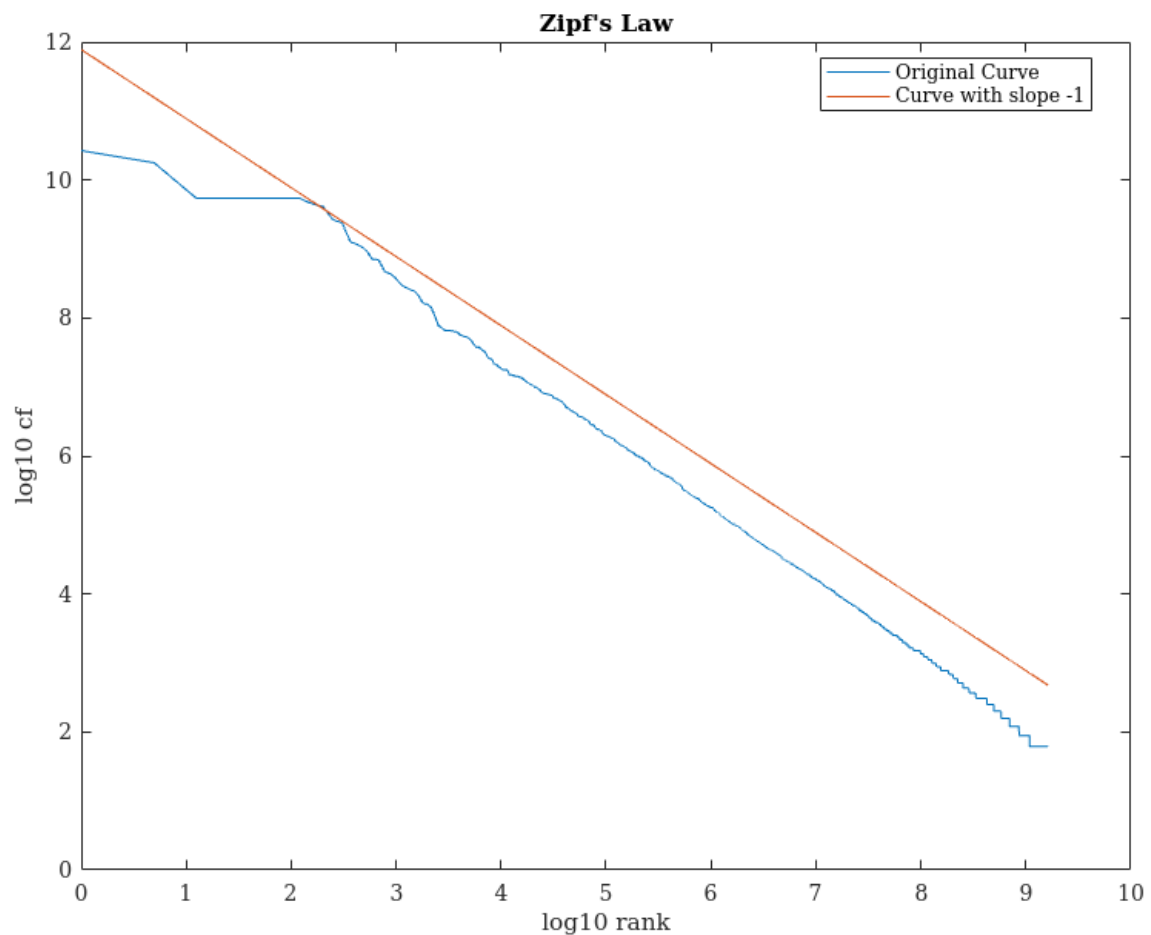
- The plot shows that '<td>' is most frequently appearing term in the corpus with the frequency of 33750.
- Most frequently appeared english word is 'to'.
- The term '**Harry**' is appeared 5657 times in the corpus.



### 0.1.3 Validating Zipf's Law

- As we know that Zipf's law is:  $cf_i \propto \frac{1}{i}$ , where  $cf_i$  is Collection Frequency of the  $i^{th}$  most common term.
- We plotted Zipf's function as  $\log cf_i = \log c - \log i$ .

- A plot with slope -1 is also shown corresponding to Zipf's function.
- Then, we applied **method of least squares** to calculate slope and intercept.
- The **intercept** of the plot is **11.8894**.
- The **slope** is **-1.1002**.
- The fit of the data to the law is not particularly good, but good enough if we compare it to the curve with slope -1.



## 0.2 EXPERIMENT -2 - BUILDING TERM-DOCUMENT MATRIX (TF-IDF)

- We have used same corpus for building Term-Document Matrix.

- Firstly, we calculated term frequency  $tf_{t,d}$  for each term  $t$  and document  $d$  and then inverse document frequency  $idf_t$  for each term  $t$ .
- Then, we calculated the weight as  $tf-idf_{t,d} = tf_{t,d} \times idf_t$

```
(base) vishal@linux:~/Desktop/IS/Experiment 13 python3 check.py
Doc 0      Doc 1      Doc 2      Doc 3      Doc 4      Doc 5      Doc 6
ABOUT      0.24303804868629444 0.8450980400142568 0.0 0.7078488453819499 0.0 1.1106565259649936 0.5440680443502757
ABSOLUTELY      0.24303804868629444 0.8450980400142568 0.0 0.7078488453819499 0.0 1.1106565259649936 0.5440680443502757
ADE      0.24303804868629444 0.8450980400142568 0.0 0.7078488453819499 0.0 1.1106565259649936 0.5440680443502757
ADVERTISE      0.24303804868629444 0.8450980400142568 0.0 0.7078488453819499 0.0 1.1106565259649936 0.5440680443502757
AFTER      0.24303804868629444 0.8450980400142568 0.0 0.7078488453819499 0.0 1.1106565259649936 0.5440680443502757
AGAIN      0.24303804868629444 0.8450980400142568 0.0 0.7078488453819499 0.0 1.1106565259649936 0.5440680443502757
AGAINST      0.24303804868629444 0.8450980400142568 0.0 0.7078488453819499 0.0 1.1106565259649936 0.5440680443502757
AGUAMENTI      0.24303804868629444 0.8450980400142568 0.0 0.7078488453819499 0.0 1.1106565259649936 0.5440680443502757
AHA      0.24303804868629444 0.8450980400142568 0.0 0.7078488453819499 0.0 1.1106565259649936 0.5440680443502757
ALBIS      0.24303804868629444 0.8450980400142568 0.0 0.7078488453819499 0.0 1.1106565259649936 0.5440680443502757
```

The result shows the Term Document matrix for first 10 term.