Type-Token Ratio (TTR)

TTR

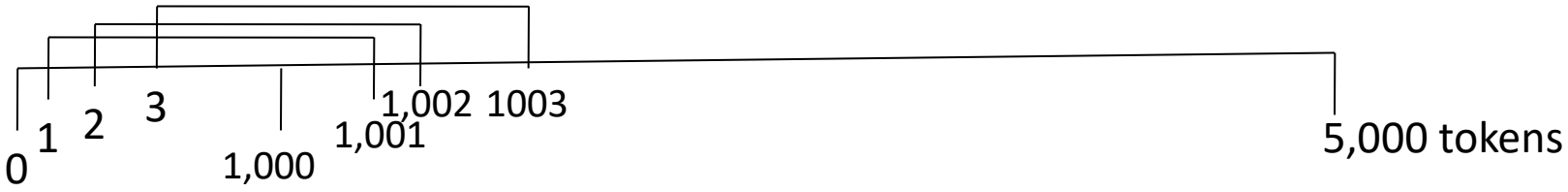
- TTR in itself is not a valid measure to find “text complexity”
- Why is it so?

TTR

- TTR in itself is not a valid measure to find “text complexity”
- Why is it so?
- If the text has more and more words, the value varies with the size of the text.

TTR

- For a valid measure, a running average is computed on consecutive 1000-word chunks of the text.



- Find TTR for 1000 word chunks across the corpus

$TTR_{0\text{-}1000}$, $TTR_{1\text{-}1001}$, $TTR_{2\text{-}1002}$, $TTR_{3\text{-}1003}$, $TTR_{4\text{-}1004}$, …

Finally, take an average:

TTR = average(TTR)

This is considered better measure for TTR

Empirical Laws

1. **Zipf's law**
   - Take a large corpus and find out what are the individual word type
   - Count the frequency of each word type in the corpus
   - List the word types in decreasing order of their frequency.

- Suppose in corpus we have words with freq like: {the(4000), an(1000), and(2000), Tom(1500),…}

Empirical Laws

- Arrange the word types in decreasing order of their frequency.

Word	Freq of word Freq
the	4000
and	2000
Tom	1500
an	1000

Empirical Laws

- Arrange the word types in decreasing order of their frequency.
- Now, **rank the word types**

Word	**Freq of word Freq**	**Rank**
the	4000	1
and	2000	2
Tom	1500	3
an	1000	4

Empirical Laws

- Arrange the word types in decreasing order of their frequency.
- Now, **rank the word types**

Word	Freq of word Freq	Rank
the	4000	1
and	2000	2
Tom	1500	3
an	1000	4

Now, what does **Zipf's law says**,

Zipf's Laws

- **Zipf's law** gives **relationship** between the **frequency of a word(f)** and **its position in the list (rank r).**
- Frequency is inversely proportional to rank:

$$f \propto \frac{1}{r}$$

- There is a constant k such that:

f * r =k

Word	Freq of word Freq	Rank	f * r
the	4000	1	4000
and	2000	2	4000
Tom	1500	3	4500
an	1000	4	4000

Zipf's Law-Probability

- Let P_r be probability of word with rank r
- N denote the total number of word occurrences

$$P_r = \frac{f_r}{N} = \frac{freq\ of\ word\ of\ rank\ r}{total\ number\ of\ tokens}$$

- Zipf's law predicts that out of a population of N elements, the frequency of elements of rank k, denoted as f (k; s, N) is:

$$f(k; s, N) = \frac{1/k^s}{\sum_{n=1}^{N} (1/n^s)}$$

where, N⟶ number of elements (corpus vocabulary size)
k ⟶ be their rank, and
s ⟶ be the value of the exponent characterizing the distribution. Set to 1 in Zipf's Law.

Zipf's Law says that there is a relationship between the probability of a word occurring in a corpus and its rank:

- Let $k_i < k_j \quad ; k_i, k_j \in Z^+$

where $k_i \rightarrow$ rank of word i , and

$k_j \rightarrow$ rank of word j

$$f(k_i; s, N) = \frac{\frac{1}{k_i^s}}{\sum_{n=1}^{N}(1/n^s)} \qquad f(k_j; s, N) = \frac{\frac{1}{k_j^s}}{\sum_{n=1}^{N}(1/n^s)}$$

$$\frac{f(k_i; s, N)}{f(k_j; s, N)} = \frac{\frac{1}{k_i^s}}{\sum_{n=1}^{N}(1/n^s)} * \frac{\sum_{n=1}^{N}(1/n^s)}{\frac{1}{k_j^s}}$$

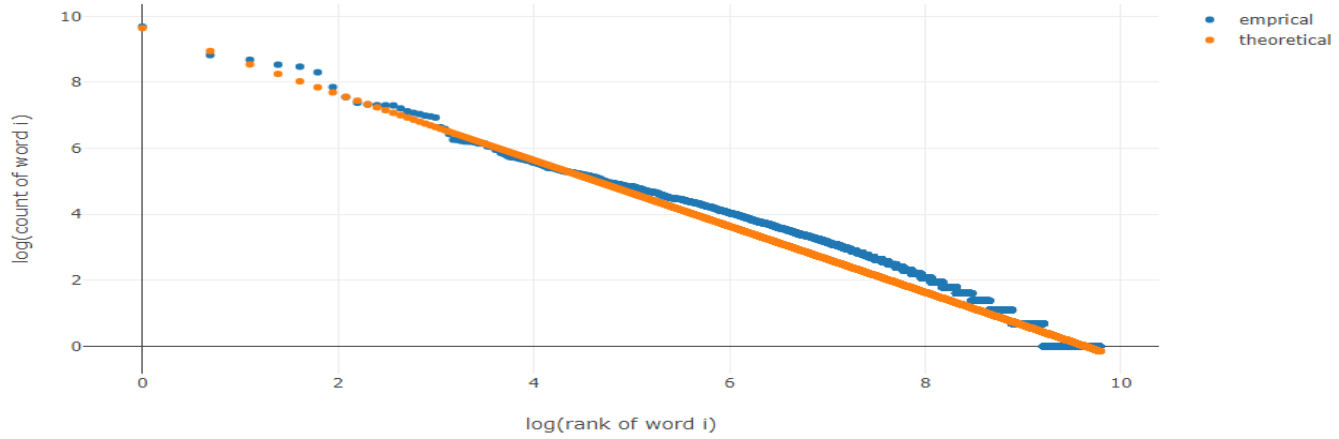
$$\frac{f(k_i;s,N)}{f(k_j;s,N)} = \frac{\frac{1}{k_i^s}}{\sum_{n=1}^{N}(1/n^s)} * \frac{\sum_{n=1}^{N}(1/n^s)}{\frac{1}{k_j^s}} = \frac{\frac{1}{k_i^s}}{\frac{1}{k_j^s}} = \left(\frac{k_i}{k_j}\right)^s$$

- Suppose s=1, and the most common word in your corpus i.e. rank 1 is “the” (i.e., word i) and the 10-th most common word i.e., rank 10 in your corpus the word “cat”(i.e., word j).
- Also, suppose that the probability of the word “the” occurring in your corpus is 0.5.
- Then according to equation 1, the probability of the word “cat” occurring in your corpus is (1/10)*0.5 = 0.05.
- In other words, for a given corpus, the relationship between the rank of a word and the probability of encountering it follows a power law.

- The “proportion” column is the number of times a particular word appears divided by the total number of words in the corpus i.e. the empirical probability.
- The “predicted_proportion” column is the theoretical probability according to Zipf’s Law with s=1. Notice that this two values are very close to each other.

	word	count	proportion	predicted_proportion
1	the	16083	0.082016	0.078348
2	of	6789	0.034621	0.039174
3	and	5885	0.030011	0.026116
4	in	5079	0.025901	0.019587
5	to	4786	0.024407	0.015670
6	a	4031	0.020556	0.013058
7	was	2575	0.013131	0.011193
8	on	1903	0.009704	0.009793
9	as	1605	0.008185	0.008705
10	that	1522	0.007762	0.007835

- Here's a log-log scatter plot showing the relationship between the rank of a word and its empirical and theoretical probability:



- Since the Zipf's Law implies a power law relationship between the rank of a word and its probability of occurring in a corpus, the shape of the theoretical line in the log-log plot will be a straight line. Notice that the empirical line more or less follows the shape of the theoretical line. Visually, this suggests that the word distribution approximately follows Zipf's Law.

Tom Sawyer- Word Distribution

Word	Freq of word Freq	Rank	f * r
the	3332	1	3332
and	2972	2	5944
a	1775	3	5325
he	877	10	8770
but	410	20	8200
be	294	30	8820
there	222	40	8880
one	172	50	8600
about	158	60	9480
never	124	80	9920
two	104	100	10400

Word	Freq of word Freq	Rank	f * r
turned	51	200	3332
you'll	30	300	5944
name	21	400	5325
comes	16	500	8770
group	13	600	8200
lead	11	700	8820
friends	10	800	8880
begin	9	900	8600
sins	2	3000	9480
could	2	4000	9920
applausive	1	5000	10400

- **f *r remains roughly the same on overall corpus.**
- **This is good evidence that Zipf's law holds.**

Zipf's other laws

- **Correlation**: Number of meanings and word frequency
- The number of meanings m of a word obeys the law: $m \propto \sqrt{f}$ $\longrightarrow$ second law

- Given the I[st] law: $f \propto \frac{1}{r}$,and

  II[nd] law: $m \propto \sqrt{f}$

Therefore, $m \propto 1/\sqrt{r}$

Heap's Law

- Gives relation between size of the vocabulary (V) and the number of tokens(N)
- Let |V| be the size of the vocabulary
- Let N be the number of tokens
- $|V| = K * N^{\beta}$
- Typically, K≈ 10 -100

  β≈ 0.4 - 06 (roughly sqroot)

Basics of Text Processing

Tokenization

- It is a process of segmenting a string of characters into words
- Categorized into:
  - Sentence segmentation
  - Word Segmentation

Challenges involved in Sentence Segmentation

- Deciding about how to mark the beginning and end of a sentence.
- Decision Criteria:
  1. Wherever there is a dot, is it end of the sentence
  2. Whether ?, !, : indicate end of the sentence
  3. Lots of blank line after dot

Other features

- Case of word with dot(.)
- Case of the word after dot(.)
- Numeric features :
  - Length of word with dot(.)
  - Prob (word with dot Occurs at EOS)
  - Prob (word after dot Occurs at EOS)

Issues in Tokenization

- Finland's capital → Finland Finlands Finland's?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??

Normalization

- Need to “normalize” terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match ***U.S.A.*** and ***USA***
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***
- Potentially more powerful, but less efficient

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***
- For sentiment analysis, MT, Information extraction
  - Case is helpful (***US*** versus ***us*** is important)