Natural Language Processing

Empirical Laws: Zipf, Mandelbrot, Heaps, Type–Token Ratio

Ramaseshan Ramachandran

Empirical Laws

WHERE THIS SESSION SITS

Before any model, count the words. It is the cheapest experiment in NLP and one of the most consequential.

- **Zipf's law** says how frequency falls with rank
- **Heaps' law** says how the vocabulary grows with the corpus
- **Type–token ratio** tries to summarise lexical richness, and fails in an instructive way

---

Every later decision in this course rests on these three curves. Stop lists, tf-idf weighting, smoothing, subword vocabularies and even the size of an LLM tokenizer are all responses to them.

TWO WORDS WE MUST NOT CONFUSE

- A **token** is a running occurrence of a word
- A **type** is a distinct word

"the cat saw the dog" has **5 tokens** and **4 types**, because *the* occurs twice.

Write N for the number of tokens and V for the number of types. Everything in this session is a statement about how N and V relate.

Brown corpus, measured with the **Assignment 1 tokenizer**: $N = 1{,}023{,}734$ tokens and $V = 44{,}324$ types. The word *the* alone is 6.84% of all tokens. The top ten words are 24.1%. And 39.5% of the types occur exactly once.

Change the tokenizer and every one of those numbers changes. Always say which one you used.

ZIPF'S LAW

This empirical law models the frequency distribution of words across languages. Zipf's law [1] states that, in a given corpus, the frequency of a word is inversely proportional to its rank in the term-frequency table.

$$f(r) = \frac{C}{r^{\alpha}}, \qquad \alpha \approx 1 \tag{1}$$

- r is the **rank** of a word, so the most frequent word has $r = 1$
- $f(r)$ is the **frequency** of the word at rank r
- α is the **decay exponent**, which controls how fast frequency falls
- C is a constant of the corpus. Setting $r = 1$ shows that $C = f(1)$, the frequency of the most common word

With $\alpha = 1$ the second word is half as frequent as the first, the tenth is a tenth as frequent, and the hundredth a hundredth.

READING THE EQUATION: A POWER LAW IS A STRAIGHT LINE

Take $\log_{10}$ of both sides of $f(r) = C\, r^{-\alpha}$:

$$\log f(r) = \log C - \alpha \log r \tag{2}$$

- This is $y = mx + c$ with $y = \log f$, $x = \log r$, slope $m = -\alpha$ and intercept $c = \log C$
- So plot frequency against rank on **log–log axes** and the law becomes a check you can do by eye
- If the points lie on a line, the law holds. The steepness of the line is α
- Fitting α is now ordinary least squares on the logged data, not curve fitting

Why this matters

Every empirical law in this session is a power law, and every power law becomes a line under logarithms. Learn the trick once and you can fit Zipf, Heaps, neural scaling laws and retrieval latency curves the same way.

ZIPF'S LAW: FREQUENCY VS RANK

Output generated from Indian budget speeches using the bash script [17] with minimal preprocessing

Word	Frequency	Rank
the	59042	1
of	46110	2
to	35873	3
and	24661	4
in	22916	5
for	14426	6
a	14366	7
is	11155	8
be	9760	9
I	9070	10
...	...	...

Word	Frequency	Rank
agricultural	479	233
important	478	234
policy	477	235
reduced	476	236
balance	476	237
...	...	...
(SFB)	1	92363
(SFAC),1	1	92364
(SFAC)	1	92365
(SEZs).,1	1	92366
...	...	...

We now fit α to exactly these fifteen numbers.

WORKED EXAMPLE, STAGE 1: THE TEMPTING SHORTCUT

Two points are enough to solve for α, so try that first. Use ranks 1 and 10:

$$\frac{f(1)}{f(10)} = \frac{C/1^{\alpha}}{C/10^{\alpha}} = 10^{\alpha} \quad \Longrightarrow \quad \alpha = \log_{10}\frac{f(1)}{f(10)}$$

$$\alpha = \log_{10}\frac{59042}{9070} = \log_{10} 6.509 = \boxed{\mathbf{0.8136}}$$

Now test it. Predict the frequency at rank 235, where the table says 477:

$$\hat{f}(235) = \frac{59042}{235^{0.8136}} = \frac{59042}{84.9} = 695$$

Predicted 695, actual 477. The estimate is 46% too high.

WORKED EXAMPLE, STAGE 2: USE EVERY POINT YOU HAVE

Least squares on $\log f$ against $\log r$, over all fifteen rows:

$$\hat{\alpha} = -\frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}, \qquad x_i = \log r_i,\ y_i = \log f_i \tag{3}$$

Method	α	C	$\hat{f}(235)$
Two points, $r = 1$ and $r = 10$	0.8136	59,042	695 (+46%)
Least squares, all 15 points	**0.9426**	**83,239**	**485** (+2%)

- The shortcut discarded thirteen measurements and paid for it
- Note that $C = 83{,}239$ is larger than the observed $f(1) = 59{,}042$. The fitted line does not pass through the top word, and it should not
- The head of the distribution is flatter than the body, so any α read off the head alone comes out too small

WORKED EXAMPLE, STAGE 3: α IS NOT A CONSTANT

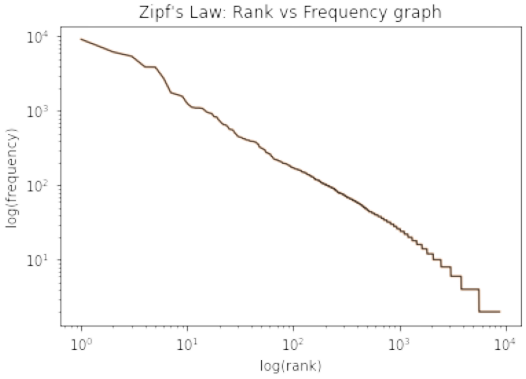
The same corpus gives a different α depending on the rank range you fit. Brown corpus, measured:

Rank range fitted	Fitted α
1…100	0.9816
1…1,000	0.9668
1…10,000	1.0726
1…44,324 (all types)	1.3492

- The tail falls faster than the head, so a wider window gives a steeper line
- An exponent quoted without its fitting range is not a measurement
- This is why the textbook claim "$\alpha \approx 1$" is safe. It is a statement about the first few thousand ranks

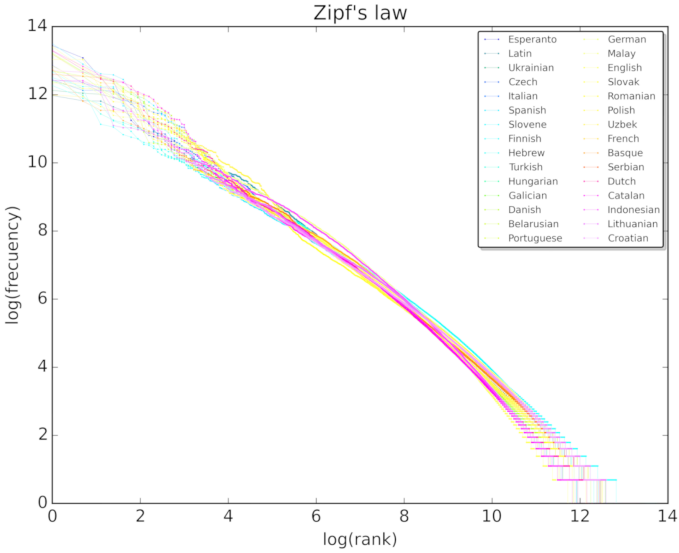
Always report the range you fitted over.

ZIPF'S LAW I



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

A plot of the rank versus frequency in a log-log scale.



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

A plot of the rank versus frequency for the first 10 million words in 30 Wikipedia (dumps from October 2015) in a log-log scale.

Source: Zipf's law -Wikipedia

FROM PROPORTIONALITY TO PROBABILITY

$f(r) = Cr^{-\alpha}$ gives counts. Divide by the total count to get a distribution over the vocabulary:

$$p(r) = \frac{r^{-\alpha}}{H_{V,\alpha}}, \qquad H_{V,\alpha} = \sum_{k=1}^{V} k^{-\alpha} \tag{4}$$

$H_{V,\alpha}$ is the **generalised harmonic number**. It is the normalising constant that makes the probabilities sum to one.

- $p(r)$ is the chance that a token drawn at random from the corpus is the rank-r word
- The **coverage** of the top k words follows immediately:

$$\text{Coverage}(k) = \sum_{r=1}^{k} p(r) = \frac{H_{k,\alpha}}{H_{V,\alpha}} \tag{5}$$

- For $\alpha \leq 1$ the sum grows without bound as V grows, which is exactly why the tail never becomes negligible

WORKED EXAMPLE: HOW MUCH TEXT DOES THE HEAD CARRY?

Brown corpus, $V = 44{,}324$, using $\alpha = 0.9668$ fitted on ranks $1 \ldots 1000$.

Top k types	% of the vocabulary	Coverage, model	Coverage, measured
10	0.02%	22.51%	24.09%
100	0.23%	41.40%	47.18%
1,000	2.26%	62.14%	69.13%
10,000	22.56%	84.56%	92.60%

- Read column two against column four. About 2% of the vocabulary carries about 69% of the running text
- The model tracks the measurement at the head and drifts at the tail, because one α cannot bend twice
- Dropping the rarest 77% of types costs you under 8% of the tokens

WHAT ZIPF'S LAW BUYS US IN THE NLP PIPELINE

- **Stop lists are short.** A few dozen words cover a quarter of every corpus, so removing them is cheap and changes the token count sharply
- **Raw counts are the wrong features.** The largest counts in a document belong to the least informative words. This is the whole motivation for inverse document frequency in the next session
- **Vocabulary truncation is nearly free in tokens.** Coverage tells you what a cut-off costs before you make it
- **Half the vocabulary is unreliable.** With 39.5% of types seen once and 54.3% seen twice or fewer, no statistic estimated per word can be trusted in the tail. This is why language models need smoothing
- **Subsampling frequent words.** word2vec discards frequent tokens with probability tied to their frequency, purely to undo the Zipfian head
- **Subword vocabularies.** BPE spends a fixed budget of merges on the head and composes the tail from pieces

Web Traffic Patterns

- A handful of sites take a large share of all visits
- The top 20% of websites account for roughly 80% of traffic
- The same shape as the Pareto principle

Caching Strategies

- CDNs exploit Zipf's law to decide what to keep hot
- A small number of pages receive most requests
- Coverage is exactly the cache hit rate you can expect

Search Algorithms

- Most queries concentrate on a small set of popular topics

Text Analytics

- Identifies stop words without a hand-written list
- Identifies the words worth focusing on
- Reduces the sample space

BASH SCRIPT FOR COMPUTING RANK AND FREQUENCY

```
# 10. Concatenate all files in TXT directory
# 11. Remove form feed character (^L)
# 12. Split into words (replace spaces with newlines)
# 13. Sort in reverse
# 14. Count unique words
# 15. Sort by frequency, descending
# 16. Add word, frequency rank
# 17. output as word_freq_rank.csv

cat xx.txt | \
tr -d '\f' | \
tr -s ' ' '\n' | \
sort -r | \
uniq -c | \
sort -nr | \
awk '{print $2 "," $1 "," NR }' \
> word_freq_rank.csv
```

MANDELBROT'S CORRECTION

Mandelbrot fits the observed distribution more closely by shifting the rank before the power is taken

$$f(r) = \frac{C}{(r+\beta)^{\alpha}} \tag{6}$$

- β moves every rank to the right, so the curve flattens where r is small and is left almost untouched where r is large
- It is a **head correction** and nothing else. Plain Zipf overestimates the very frequent words
- β sits inside the logarithm, so it cannot be found by least squares. Scan β, and least-squares α and C at each value
- How large β comes out depends almost entirely on **how much of the tail you fit**, as the next frame shows

WORKED EXAMPLE: DOES β EARN ITS PLACE?

Same fifteen budget-speech rows, same procedure, one extra parameter.

Model	α	β	Squared error	$\hat{f}(235)$
Zipf	0.9426	0	0.0453	485
Mandelbrot	0.9843	0.5	**0.0302**	**472**

Actual $f(235) = 477$. The error drops by a third for one extra parameter.

The warning that comes with it

The valley is shallow, so many pairs fit almost as well:

β	best α	squared error
0.0	0.9426	0.0453
0.5	0.9843	0.0302
1.0	1.0183	0.0356
2.0	1.0749	0.0643

Report (α, β) as a pair. Neither number means anything alone.

WORKED EXAMPLE: β IS SET BY THE FITTING RANGE

The budget example gave $\beta = 0.5$, which looks negligible. Fit Brown instead, and vary how much of the tail you include:

Rank range fitted	α	β
1…1,000	0.9668	**0.00**
1…10,000	1.0984	10.25
1…44,324 (all ranks)	1.5180	**403.25**

- Fit the head alone and β is exactly zero, so Mandelbrot buys nothing. Plain Zipf already describes the head
- Fit every rank, which is what the Assignment 1 grader does, and β runs into the hundreds
- A large β is not a red flag about your corpus. It is what a single power law needs in order to cover a curve that *bends*
- α and β rise together, because both are absorbing the same bend

Quote α, β and the rank range. All three, or none.

HERDAN OR HEAPS' LAW

Zipf's law describes a frozen corpus. Heaps' law [2] describes what happens as a corpus *grows*.

Let N be the number of tokens read so far and $V(N)$ the number of distinct types seen. Then

$$\begin{aligned} V(N) &\propto N^b \\ &= K N^b \end{aligned} \tag{7}$$

- K is a scale constant, typically $10 \leq K \leq 100$ on raw English text
- b is the growth exponent, typically $0.4 \leq b \leq 0.6$ for English
- $b < 1$ is the whole content of the law. New words keep arriving, at a diminishing rate
- Those ranges hold near a million tokens. Smaller corpora break both

---

Setting $N = 1$ gives $V = K$, so K is the vocabulary size the fitted curve claims after one token. It is a fitting constant, not a quantity with a meaning.

READING THE EQUATION

Take logarithms again:

$$\log V(N) = \log K + b \log N \tag{8}$$

- A straight line of slope b on log–log axes, exactly as with Zipf
- b is an **elasticity**. Doubling the corpus multiplies the vocabulary by 2^b
- With $b = 0.5$, four times the text gives twice the vocabulary
- The function has **no asymptote**. There is no corpus size at which $V(N)$ stops growing

---

Why this matters

However much text you collect, the next document can still hold a word you have never seen. A name, a typo, a coinage, a hashtag, a chemical formula. **The vocabulary of a natural language is not a closed set.** Any system built around a fixed word list has already decided to fail on the tail.

WORKED EXAMPLE: FITTING b ON THE BROWN CORPUS

Sample the vocabulary every 1,000 tokens, exactly as the Assignment 1 grader does, and fit all 1,024 points. Five of them:

N tokens	V types	V/N
1,000	468	0.4680
10,000	2,584	0.2584
100,000	12,962	0.1296
500,000	31,854	0.0637
1,000,000	43,648	0.0436

$$b = \boxed{\mathbf{0.5764}} \qquad k = \boxed{\mathbf{16.18}}$$

- The corpus multiplied by 1000, the vocabulary by only 93
- Note the third column falling. That column is the type–token ratio, and we return to it shortly

WORKED EXAMPLE: WHAT EXTRAPOLATION COSTS

Now fit the same law using only the first 100,000 tokens, then predict a corpus ten times larger.

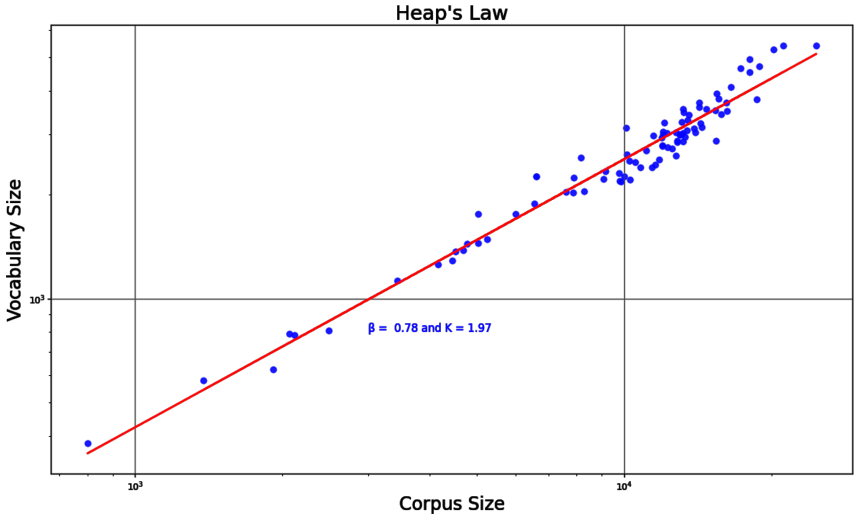
Fitted on	b	k
$N \leq 100{,}000$	0.7176	3.50
the full million	0.5764	16.18

$$\hat{V}(10^6) = 3.50 \times (10^6)^{0.7176} = 70{,}799 \qquad \text{actual } 43{,}648$$

One decade of extrapolation, 62% too many types.

- b is not a constant of the language either. It drifts down as N grows, because the log–log curve is gently concave
- The error is an *over*-estimate, which is the safe direction for memory planning and the wrong direction for cost estimates
- Fit over the range you will actually operate in

HEAPS' LAW


Heap's Law
Vocabulary Size
Corpus Size
β = 0.78 and K = 1.97
10³
10³
10⁴

ZIPF AND HEAPS ARE ONE FACT SEEN TWICE

If frequency falls as $r^{-\alpha}$, a word at rank r needs roughly r^{α} tokens of text before it appears at all. Reading that backwards, N tokens reveal about $N^{1/\alpha}$ types, so

$$b \approx \frac{1}{\alpha} \qquad (\alpha > 1) \tag{9}$$

Brown α, fitted over all ranks	1.3492
predicted $b = 1/\alpha$	0.7412
measured b	0.5764

- Right order of magnitude, wrong decimals, and the gap is honest
- The derivation assumes one α covers every rank, and we saw α move from 0.98 to 1.35 across ranges
- Take the relationship, not the number. Steeper decay means fewer rare words, which means slower vocabulary growth

WHICH KNOB MOVES TOKENS, WHICH MOVES TYPES

Measured on Brown with the Assignment 1 tokenizer. Each row is a share of the raw figures.

Operation	tokens left	types left
Stop words removed	**55.5%**	99.8%
Numbers stripped	99.0%	97.5%
Porter stemming	100.0%	**65.4%**
Drop count ≤ 5	94.1%	**28.0%**

- Stop-word removal takes 44% of the tokens and 0.2% of the types. Zipf's law says exactly this
- Stemming is the mirror image. Every token stays, and 35% of the vocabulary collapses
- Dropping every type seen five times or fewer removes 72% of the vocabulary for 5.9% of the tokens
- The Heaps fit barely moves: k goes $16.18 \to 17.33$ and b goes $0.5764 \to 0.5958$

Stop-word removal attacks N. Truncation and stemming attack V.

WHAT HEAPS' LAW BUYS US IN THE NLP PIPELINE

- **Index and memory planning.** An embedding table costs $|V| \times d$ floats. Heaps' law is how you size it before you collect the data
- **Out-of-vocabulary items are permanent.** Not a corner case to patch, but the normal condition of any word-level system
- **Fixed-size subword vocabularies.** BPE, WordPiece and SentencePiece exist to escape $V(N) = KN^b$ altogether. A closed set of pieces composes an open set of words
- **Softmax cost.** The output layer of a language model is $O(|V|)$ per token. Unbounded V is unbounded cost, which is what hierarchical softmax and negative sampling attack
- **Data scaling.** The same log–log fitting you did here is how neural scaling laws are read

IMPLICATIONS FOR NLP

Predictive Power

Estimate vocabulary and coverage before collecting the data.

Corpus Analysis

Compare corpora on the same axes, at the same size.

Model Development

Guides vocabulary sizing, truncation and smoothing.

TYPE–TOKEN RATIO (TTR)

A natural summary of lexical richness is the ratio of types to tokens:

$$\text{TTR} = \frac{V}{N} = \frac{\text{number of distinct words}}{\text{total number of words}} \tag{10}$$

- TTR lies in $(0, 1]$. A text that never repeats a word scores 1
- Higher TTR is read as a richer vocabulary, and lower as repetitive or formulaic text
- *"the cat saw the dog"* gives $\text{TTR} = 4/5 = 0.8$
- Brown corpus as a whole gives $44{,}324/1{,}023{,}734 = 0.0433$

Those two numbers are not comparable, and the reason is the point of this section.

HEAPS' LAW TELLS US TTR MUST FALL

Substitute $V = KN^b$ into $TTR = V/N$:

$$TTR(N) = \frac{KN^b}{N} = KN^{b-1} \tag{11}$$

Since $b < 1$, the exponent $b - 1$ is negative, so TTR **falls as the text grows**. Brown's fit, $k = 16.18$ and $b = 0.5764$:

N	predicted TTR	measured TTR
1,000	0.8673	0.4680
10,000	0.3270	0.2584
100,000	0.1233	0.1296
1,000,000	0.0465	0.0436

Same author, same style, four different answers.

TTR is not a property of a text. It is a property of a text *at a length*.

Note the first row. The fit is good near a million tokens and poor at a thousand, because the curve is concave and 1,024 points pull the line towards the large-N end. A fitted law summarises a curve; it is not an oracle for each point.

WORKED EXAMPLE: THE LENGTH TRAP IN ACTION

Six genres of the Brown corpus. Raw TTR uses the whole genre. Standardised TTR averages over non-overlapping 8,000-token windows.

Genre	N	raw TTR	rank	std. TTR	rank
press reportage	90,089	0.1359	3	**0.3077**	**1**
humour	18,450	**0.2552**	**1**	0.3021	2
general fiction	58,605	0.1458	2	0.2744	3
romance	58,966	0.1307	4	0.2520	4
learned	164,036	0.0885	6	0.2478	5
government	63,214	0.1106	5	0.2320	6

- Raw TTR crowns humour, which is also the shortest genre at 18,450 tokens. The measurement is reading length
- Raw TTR ranks fiction above press reportage. On equal windows the order reverses
- Learned prose looks worst on raw TTR and is mid-table on equal windows. It is long, not poor

LENGTH-CORRECTED ALTERNATIVES

- **Standardised TTR (MSTTR).** Cut the text into equal windows of fixed size W, compute TTR in each, and average:

$$\text{MSTTR} = \frac{1}{m} \sum_{j=1}^{m} \frac{V_j}{W}$$

  Simple, and the one used in the table above. Its weakness is that the answer depends on W
- **Root TTR.** $V/\sqrt{N}$, which is Heaps' law with b forced to 0.5. It only removes the length effect if b really is 0.5
- **Report (k, b) instead.** The honest summary of lexical richness is the fitted Heaps curve, not a single ratio. Quote the pair, because k and b are coupled

---

Never compare type–token ratios across samples of different sizes.

APPLICATIONS OF TTR

- Monitor vocabulary usage in a text or by a writer
- Track child vocabulary development, where sample size is controlled by design
- Estimate vocabulary variation across genres or authors
- Detect templated or repetitive machine-generated text. A decoder stuck in a loop shows a collapsing TTR
- Diagnose degenerate decoding. Low-temperature and greedy generation both drive TTR down, which is a symptom you will meet again in session 19

EXERCISES

1. Fit α on your own corpus by least squares over ranks $1 \ldots 100$, then over all ranks. Report both, with the ranges. Explain the difference in one sentence.
2. Refit with Mandelbrot's β and report (α, β) as a pair. Show that a range of pairs fits almost equally well.
3. Compute the coverage of the top 10, 100 and 1000 words directly from your counts. Compare with the harmonic-number prediction using your fitted α.
4. Fit Heaps' law on the first tenth of your corpus, predict V for the whole, then measure it. Report the signed error.
5. Find the percentage of tokens that are stop words, and the percentage of *types* that are stop words. Explain why the two numbers are so far apart.
6. Compare the TTR of two genres, first raw and then on equal windows. Report whether the ranking changes.

Predict every number before you compute it. The gap between prediction and measurement is where the marks are.

REFERENCES I

[1] George Kingsley Zipf. "The psychology of language". In: *Encyclopedia of psychology*. Philosophical Library, 1946, pp. 332–341.

[2] Kim Chol-jun. *Proper Interpretation of Heaps' and Zipf's Laws*. 2025. arXiv: 2305.15413 [physics.soc-ph]. URL: https://arxiv.org/abs/2305.15413.