3: Statistical Properties of Language
Machine Learning and Real-world Data (MLRD)

Simone Teufel
Last session: You implemented a Naive Bayes classifier

- Smoothed vs Unsmoothed
- The accuracy of the un-smoothed classifier was seriously affected by unseen words.
- We implemented add-one (Laplace) smoothing:

\[ \hat{P}(w_i|c) = \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)} = \frac{\text{count}(w_i, c) + 1}{(\sum_{w \in V} \text{count}(w, c)) + |V|} \]

- Smoothing helped!
Today: frequency distributions in language

Questions:

- Why did smoothing help? (or in other words:)
- What is it about the distribution of words in a language that affected the performance of the un-smoothed classifier?
- Two Laws: Zipf’s Law and Heap’s Law
Zipf’s Law: Word frequency distributions obey a power law

- There are a small number of very high-frequency words
- There are a large number of low-frequency words
- Zipf’s law: the $n$th most frequent word has a frequency proportional to $1/n$

“a word’s frequency in a corpus is inversely proportional to its rank”
The parameters of Zipf’s law are language-dependent

Zipf’s law:

\[ f_w \approx \frac{k}{r_w^\alpha} \]

where

\( f_w \): frequency of word \( w \)
\( r_w \): frequency rank of word \( w \)
\( \alpha, k \): constants (which vary with the language)

e.g. \( \alpha \) is around 1 for English but 1.3 for German
The parameters of Zipf’s law are language-dependent

Actually...

\[ f_w \approx \frac{k}{(r_w + \beta)^\alpha} \]

where

\( \beta \): a shift in the rank

see summary paper by Piantadosi

https://link.springer.com/article/10.3758/s13423-014-0585-6

we won’t worry about the rank-shift today
There are a small number of high-frequency words...

Moby Dick has 206,052 words in total.
Similar sorts of high-frequency words across languages

Top 10 most frequent words in some large language samples:

- **English**: the (6,187,267), of (2,941,444), and (2,682,863), a (2,126,369), to (1,620,850), is (998,389), was (923,948), was (923,948), to (917,579)
- **German**: der (7,377,879), die (7,036,092), der (7,377,879), die (7,036,092)
- **Spanish**: que (32,894), de (32,116), de (32,116)
- **Italian**: non (25,757), di (22,868), di (22,868)
- **Dutch**: de (4,770), van (2,259), van (2,259)
Similar sorts of high-frequency words across languages

Top 10 most frequent words in some large language samples:

English

1. the 6,187,267
2. of 2,941,444
3. and 2,682,863
4. a 2,126,369
5. in 1,812,609
6. to 1,620,850
7. it 1,089,186
8. is 998,389
9. was 923,948
10. to 917,579

BNC, 100Mw
Similar sorts of high-frequency words across languages

Top 10 most frequent words in some large language samples:

<table>
<thead>
<tr>
<th>Rank</th>
<th>English</th>
<th>Frequency</th>
<th>Rank</th>
<th>German</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>6,187,267</td>
<td>1</td>
<td>der</td>
<td>7,377,879</td>
</tr>
<tr>
<td>2</td>
<td>of</td>
<td>2,941,444</td>
<td>2</td>
<td>die</td>
<td>7,036,092</td>
</tr>
<tr>
<td>3</td>
<td>and</td>
<td>2,682,863</td>
<td>3</td>
<td>und</td>
<td>4,813,169</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>2,126,369</td>
<td>4</td>
<td>in</td>
<td>3,768,565</td>
</tr>
<tr>
<td>5</td>
<td>in</td>
<td>1,812,609</td>
<td>5</td>
<td>den</td>
<td>2,717,150</td>
</tr>
<tr>
<td>6</td>
<td>to</td>
<td>1,620,850</td>
<td>6</td>
<td>von</td>
<td>2,250,642</td>
</tr>
<tr>
<td>7</td>
<td>it</td>
<td>1,089,186</td>
<td>7</td>
<td>zu</td>
<td>1,992,268</td>
</tr>
<tr>
<td>8</td>
<td>is</td>
<td>998,389</td>
<td>8</td>
<td>das</td>
<td>1,983,589</td>
</tr>
<tr>
<td>9</td>
<td>was</td>
<td>923,948</td>
<td>9</td>
<td>mit</td>
<td>1,878,243</td>
</tr>
<tr>
<td>10</td>
<td>to</td>
<td>917,579</td>
<td></td>
<td>sich</td>
<td>1,680,106</td>
</tr>
</tbody>
</table>

BNC, 100Mw
“Deutscher Wortschatz”, 500Mw
Similar sorts of high-frequency words across languages

Top 10 most frequent words in some large language samples:

<table>
<thead>
<tr>
<th>English</th>
<th>German</th>
<th>Spanish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 the</td>
<td>1 der</td>
<td>1 que</td>
</tr>
<tr>
<td>2 of</td>
<td>2 die</td>
<td>2 de</td>
</tr>
<tr>
<td>3 and</td>
<td>3 und</td>
<td>3 no</td>
</tr>
<tr>
<td>4 a</td>
<td>4 in</td>
<td>4 a</td>
</tr>
<tr>
<td>5 in</td>
<td>5 den</td>
<td>5 la</td>
</tr>
<tr>
<td>6 to</td>
<td>6 von</td>
<td>6 el</td>
</tr>
<tr>
<td>7 it</td>
<td>7 zu</td>
<td>7 es</td>
</tr>
<tr>
<td>8 is</td>
<td>8 das</td>
<td>8 y</td>
</tr>
<tr>
<td>9 was</td>
<td>9 mit</td>
<td>9 en</td>
</tr>
<tr>
<td>10 to</td>
<td>10 sich</td>
<td>10 lo</td>
</tr>
</tbody>
</table>

BNC, 100Mw

“Deutscher Wortschatz”, subtitles, 27.4Mw
Similar sorts of high-frequency words across languages

Top 10 most frequent words in some large language samples:

<table>
<thead>
<tr>
<th></th>
<th>English</th>
<th>German</th>
<th>Spanish</th>
<th>Italian</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>der</td>
<td>que</td>
<td>non</td>
</tr>
<tr>
<td>2</td>
<td>of</td>
<td>die</td>
<td>de</td>
<td>di</td>
</tr>
<tr>
<td>3</td>
<td>and</td>
<td>und</td>
<td>no</td>
<td>che</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>in</td>
<td>a</td>
<td>è</td>
</tr>
<tr>
<td>5</td>
<td>in</td>
<td>den</td>
<td>la</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>to</td>
<td>von</td>
<td>el</td>
<td>la</td>
</tr>
<tr>
<td>7</td>
<td>it</td>
<td>zu</td>
<td>es</td>
<td>il</td>
</tr>
<tr>
<td>8</td>
<td>is</td>
<td>das</td>
<td>y</td>
<td>un</td>
</tr>
<tr>
<td>9</td>
<td>was</td>
<td>mit</td>
<td>en</td>
<td>a</td>
</tr>
<tr>
<td>10</td>
<td>to</td>
<td>sich</td>
<td>lo</td>
<td>per</td>
</tr>
</tbody>
</table>

**BNC, 100Mw**

“Deutscher Wortschatz”, 500Mw

subtitles, 27.4Mw

subtitles, 5.6Mw
Similar sorts of high-frequency words across languages

Top 10 most frequent words in some large language samples:

<table>
<thead>
<tr>
<th></th>
<th>English</th>
<th>German</th>
<th>Spanish</th>
<th>Italian</th>
<th>Dutch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>der</td>
<td>que</td>
<td>non</td>
<td>de</td>
</tr>
<tr>
<td>2</td>
<td>of</td>
<td>die</td>
<td>de</td>
<td>di</td>
<td>en</td>
</tr>
<tr>
<td>3</td>
<td>and</td>
<td>und</td>
<td>no</td>
<td>che</td>
<td>het/t</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>in</td>
<td>a</td>
<td>è</td>
<td>van</td>
</tr>
<tr>
<td>5</td>
<td>in</td>
<td>den</td>
<td>la</td>
<td>e</td>
<td>ik</td>
</tr>
<tr>
<td>6</td>
<td>to</td>
<td>von</td>
<td>el</td>
<td>la</td>
<td>te</td>
</tr>
<tr>
<td>7</td>
<td>it</td>
<td>zu</td>
<td>es</td>
<td>il</td>
<td>dat</td>
</tr>
<tr>
<td>8</td>
<td>is</td>
<td>das</td>
<td>y</td>
<td>un</td>
<td>die</td>
</tr>
<tr>
<td>9</td>
<td>was</td>
<td>mit</td>
<td>en</td>
<td>a</td>
<td>in</td>
</tr>
<tr>
<td>10</td>
<td>to</td>
<td>sich</td>
<td>lo</td>
<td>per</td>
<td>een</td>
</tr>
</tbody>
</table>

BNC, 100Mw “Deutscher Wortschatz”, 500Mw subtitles, 27.4Mw subtitles, 5.6Mw subtitles, 800Kw
It is helpful to plot Zipf curves in log-space


By fitting a simple line to the data in log-space we can estimate the language specific parameters $\alpha$ and $k$ (we will do this today!)
In log-space we can more easily estimate the language specific parameters.

From Piantadosi [https://link.springer.com/article/10.3758/s13423-014-0585-6]
Zipfian (or near-Zipfian) distributions occur in many collections

- Sizes of settlements
- Frequency of access to web pages
- Size of earthquakes
- Word senses per word
- Notes in musical performances
- Machine instructions
- ...
Zipfian (or near-Zipfian) distributions occur in many collections.
There is also a relationship between vocabulary size and text length.

So far we have been thinking about frequencies of particular words:

- we call any unique word a **type**: *the* is a word type
- we call an instance of a type a **token**: there are 13721 *the* tokens in *Moby Dick*
- the number of types in a text is the size of the vocabulary (also called dictionary)

Today you will also explore this relationship.
Heaps’ law describes the relationship between vocabulary and text-length

Heaps’ Law:
The relationship between the size of a vocabulary and the size of text that gave rise to it is

\[ u_n = kn^\beta \]

where

- \( u_n \): number of types (unique items), i.e. vocabulary size
- \( n \): total number of tokens, i.e. text size
- \( \beta, k \): constants (language-dependent)
  - \( \beta \) is around \( \frac{1}{2} \)
  - \( 30 \leq k \leq 100 \)
Heaps’ Law

- No saturation: there will always be more new types
- As we progress through a text it takes longer and longer to encounter a new type
It is helpful to plot Heaps’ law in log-space
Zipf’s law and Heaps’ law affected our classifier

- The Zipfian curve has a lot of probability mass in the long tail.
- By Heaps’ law, we need increasing amounts of text to see new word types in the tail.

![Graph showing the relative frequency in Moby Dick as a function of rank. The graph is a plot with the x-axis labeled 'Rank' and the y-axis labeled 'Relative frequency in Moby Dick.' The data points are distributed such that the frequency decreases rapidly at the beginning and then tapers off towards the tail.]
Zipf’s law and Heaps’ law affected our classifier

- With MLE, only seen types receive a probability estimate:
  e.g. we used:
  \[
  \hat{P}_{MLE}(w_i|c) = \frac{\text{count}(w_i, c)}{\sum_{w \in V_{training}} \text{count}(w, c)}
  \]

- True probability (e.g. for NEG class): orange; MLE: blue
- Total probabilities must sum to 1; in MLE all that probability mass is given to seen types
- MLE overestimates the probability of seen types (as opposed to unseen)
Smoothing redistributes the probability mass

- Add-one smoothing redistributes the probability mass.
  e.g. we used:

\[
\hat{P}(w_i|c) = \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V}(\text{count}(w, c) + 1)} = \frac{\text{count}(w_i, c) + 1}{(\sum_{w \in V}\text{count}(w, c)) + |V|}
\]

- It takes some portion away from the MLE overestimate.
- It redistributes this portion to the unseen types.
- Better estimate; still not perfect.
Today we will investigate Zipf’s and Heaps’ law in movie reviews

Follow task instructions on moodle to:

- Plot a frequency vs rank graph for larger set of movie reviews (you are given chart plotting code)
- Plot a log frequency vs log rank graph
- Indicate the location of your 10 chosen words from Tick 1, e.g. in colour, on this plot.
- Use least-squares algorithm to fit a line to the log-log plot (you are given best-fit code)
- Estimate the parameters of the Zipf equation
- Plot type vs token graph for the movie reviews
Ticking for Task 3

There is no automatic ticker for Task 3

- Write everything in your lab book
- Save all your graphs (as screenshots or otherwise)