Vector Semantics: Lecture 2

András Kornai
SZTAKI Computer Science Research Institute

21 February 2024

Word frequency distributions

- First, collect a **corpus** reflective of the language variety that you care about (e.g. the language of Dickens, the language of neurobiology, or the language of "the web")
- How do you make sure the corpus is representative? Easy for Dickens, harder for neurobiology, very hard for the web
- Next, you **tokenize**. Key issues: do you equate lowercase and uppercase versions? How do you treat punctuation? Where do you draw the word boundaries?
- Finally: do you want to **lemmatize**? Are *represent* and *represents* the same word as *representing* and *represented*? How about *representative* and *representation*?
- In English, the issue can be largely avoided, but in many languages it can't be
- We will start with English, but morphology will stay on the agenda

Corpora, tokenization

- HLT has plenty, ask and you shall receive (see `Corpora` after the overview)
- You can also use some preexisting crawler, or undertake to dust off our own, see `Resources/wac4.pdf`
- We have fast C code for tokenization, but for many goals standard unix utilities are already sufficient (see `Resources/bentley_1986.pdf`)
- Spot querying by ordinary (linux) means
- Tokenizing is also easy in English with `sed`. For Hungarian `https://github.com/nytud/quntoken` is fine
- Morphological analysis is much more complex, for English we may start using EMOR which is based on SFST (München has Latin, German, Turkish, and Malayalam morphologies for the fun-loving – HLT also has corpora for most of these)

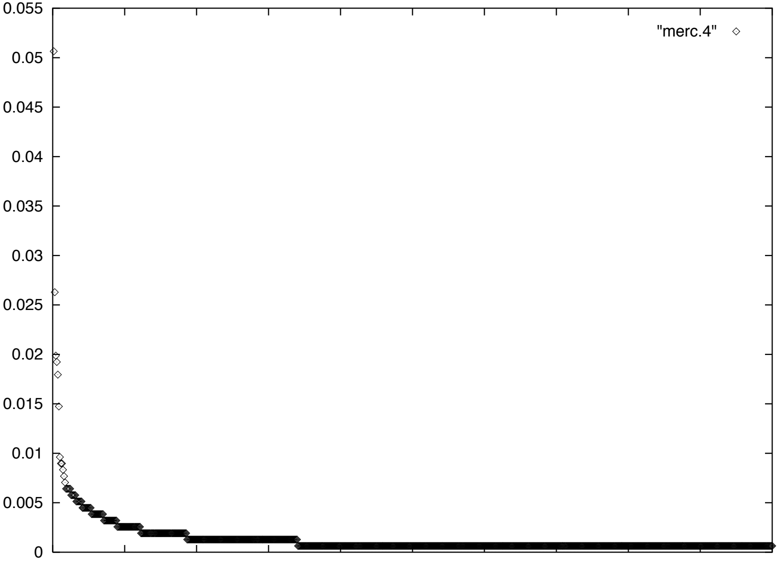
Some notation

- We begin with a probability distribution of which a corpus C is merely a sample. We adopt this view even for 'closed' corpora such as the works of Dickens, for a new manuscript can always surface, and our interest is in the population (e.g. the language of Dickens, the language of neurobiology, etc)
- For each type w we obtain $F_C(w)$ (called the *sample counts*). HLT has counting tools that are considerably faster than code you could write on the spot
- We denote corpus size $|C| = \sum F_C(w)$ by N, and consider *corpus frequency* $f_C(w) = F_C(w)/N$. We are interested in $f_C(w)$ only as an estimate of $p(w)$, the probability of w in the population
- We denote by $V(C)$ the number of different *types* in the corpus. To the extent we draw different samples C and D from the same population we find that V depends heavily on $|C|$ but only minimally on the choice of C itself, so we will write $V(N)$

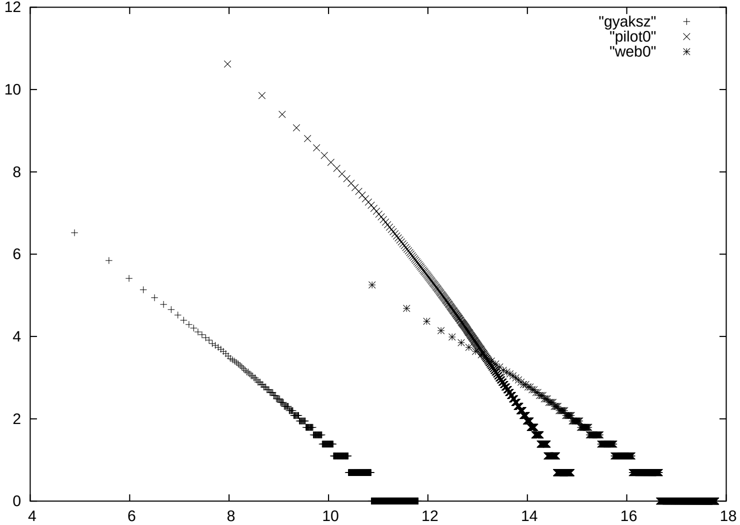
Easily repeatable observations

- We notice that empirical frequencies converge very slowly, and give a weak estimate of the probability for the sample sizes (N below a few million) common until the 1980s. In practice, not even the top 10 words show stable frequencies below $N = 10^9$ (gigaword corpora)
- We also notice that empirical frequencies span as many orders of magnitude as the corpus size permits. There are always *hapax legomena*, words that appear with $F = 1$, $f = 1/N$ (see `Resources/indra.pdf`)
- To reduce the effect of slow convergence, we rearrange the data by decreasing frequency. The most frequent word (in English *the*) will be considered 'rank 1', the 2nd most frequent 'rank 2' and so on. Ranks r are between 1 and $V(N)$, and instead of $p(w)$ we will look at p_r.
- We plot p_r as a function of r on a log-log scale

Direct plot (no log-log transform)

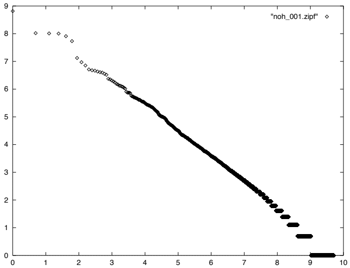


More standard plots



Normalization

- We normalize the x (log rank) axis by scaling with $\log V(n)$ so that our normalized x is always in the [0-1] range, no matter how big the corpus
- The rearrangement by rank automatically makes the function monotonically decreasing

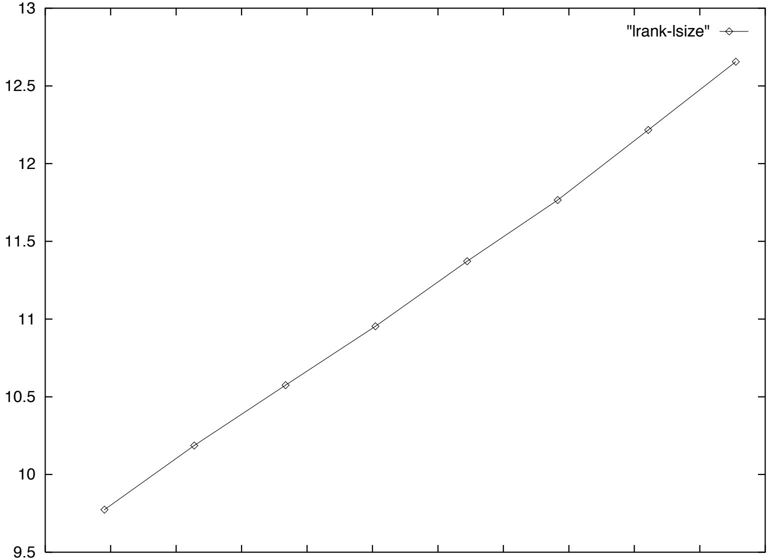


- Zipf fit a linear curve on the log-log plot, which means $\log(F(x)) \sim H_C - B_C x \log V(N)$

Let's look at the intercepts first

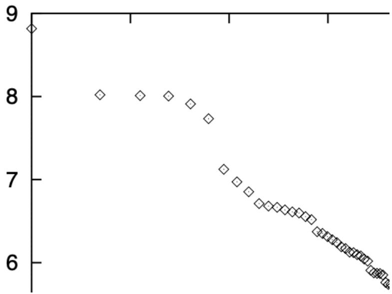
- Close to $x = 0$. Here we have $\log F_1$ (the log count of the most frequent item). Consider English *the* which takes up about 5% of the corpus, this means $\log 0.05N \sim H_C - B_C/V(N)$ Since $V(N) \to \infty$ with $N \to \infty$ whereas B_C remains bounded (close to 1), we have $H_C \sim \log N + \log p_1$ and the frequency of the most frequent word is constant, so H_C is about $\log N$.
- Close to $x = 1$. At the highest (log) rank we see a hapax, so we have $\log(F_{V(N)}) = \log(1) = 0$ which gives the linear equation $H_C = B_C \log V(N)$, so by plugging in our H_C estimate we get $\log(N)/\log(V(N)) = B_C$
- Taking $q = 1/B$ we have $V(N) = N^q$ known as *Herdan's law of vocabulary growth*.

Herdan's Law



In the high range

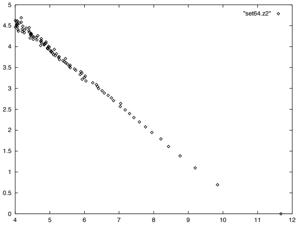
- The linearity of the empirical curve is highly questionable:



- To get rid of the problem, we assume an urn model with two urns, roughly corresponding to *function words* and *content words*. We assign the top k words to the first urn, accounting for maybe as much as 50% of the probability mass
- For a broader overview see `mitzenmacher_2003.pdf`, `hmwat.pdf`

In the low range

- The fit is much better! Let c_1 be the number of hapax legomena, c_2 the number of dis legomena, etc. *Zipf's Second Law* aka 'number-frequency law' says that plotting $\log n$ against $\log c_n$ will be linear, with slope $\sim -1/2$



- Theorem 3 (Kornai, 1999a) If a distribution satisfies Zipf's Law with slope parameter B, it satisfies Zipf's Second law with parameter $D = B/(1+B)$
- Second Law $\not\Rightarrow$ First Law (For nice clean Tauberian fun see `zipf.pdf`)

Entropy

- Character entropy (finitely many choices) is actually very important (`cfreq` tool for character counts) but we will really concentrate on word entropy here
- We cut the sum in two parts at some boundary k. To get the two urns roughly equal, $P_k = \sum_{i=1}^{k} p_i \sim \sum_{i=k+1}^{\infty} p_i$, takes about 256 words for English, 4096 (unstemmed) for Hungarian. Altogether, we have

$$1-P_k = C_k \sum_{r=k+1}^{N^{1/B}} r^{-B} \approx C_k \int_k^{N^{1/B}} x^{-B} dx = \frac{C_k}{(1-B)} [N^{\frac{1-B}{B}} - k^{1-B}]$$

where C_k is some constant of proportionality that guarantees that the probailities sum to 1. This yields $C_k \approx (1 - P_k)(B - 1)k^{B-1}$. Away from the very top of the range, the Zipf fit is very good, so the computation is not very sensitive to the choice of k.

Word entropy con't

- We have

$$H = -\sum_{r=1}^{k} p_r \log_2(p_r) - \sum_{r=k+1}^{N^{1/B}} p_r \log_2(p_r)$$

We use direct entropy computation for the "function word" urn, and use Zipf's laws for the "content word" urn. For English, P_{256} is about 0.52, and $B \sim 1.25$, for Hungarian $P_{4096} \sim 0.5$.

-

$$H \approx H_k + \frac{1 - P_k}{\log(2)}(B/(B-1) - \log(B-1) + \log(k) - \log(1-P_k))$$

This yields H=12.67 bits for English, H=15.41 bits for Hungarian.

Borbély, Gábor and András Kornai (June 2019). "Sentence Length". In: *Proceedings of the 16th Meeting on the Mathematics of Language*. Toronto, Canada: Association for Computational Linguistics, pp. 114–125. URL: https://www.aclweb.org/anthology/W19-5710.

Kornai, András (2002). "How many words are there?" In: *Glottometrics* 2.4, pp. 61–86.

— (1999a). "Zipf's law outside the middle range". In: *Proceedings of the Sixth Meeting on Mathematics of Language*. Ed. by J. Rogers. University of Central Florida, pp. 347–356.

Mitzenmacher, Michael (2003). "A Brief History of Generative Models for Power Law and Lognormal Distributions". In: *Internet Mathematics* 1.2, pp. 226–251.