# Language Modeling (with n-grams)

Chapter 4 J&M'09

# *Language Models*

**Probabilistic language modeling**

GOAL: determine the probability $P$ of a sequence of words

Applications:

- Part of speech (POS) tagging
  $P(\text{she}_{PRN}\ \text{bikes}_{V-3SG}) > P(\text{she}_{PRN}\ \text{bikes}_{CN-PL})$
- Spelling correction
  $P$(their cat is sick) $>$ P(there cat is sick)
- Speech recognition
  $P$(I can forgive you) $>$ $P$(I can for give you)
  $P$(I can forget you) $>$ $P$(I can for get you)
- Etc.: machine translation, language identification, authorship identification, word similarity, predictive text input, etc.

# *Language Models*

**General goal:**

- Compute the joint probability of a word sequence $W$
  $P(W) = P(w_1, w_2, ..., w_n)$

  (note: sometimes $w_1^n$ is used to abbreviate $w_1, w_2, ..., w_n$ )

- Related task: probability of an upcoming word
  $P(w_n|w_1, w_2, ..., w_{n-1})$

# *Language Models*

What is probability of '*the*' following '*It is easy to see*' ?
More formally, what is $P(\textit{the} \mid \textit{It is easy to see})$?

**Approach #1:** count ratio.

$$P(\textit{it is easy to see the}) = \frac{C(\textit{it is easy to see the})}{C(\textit{it is easy to see})}$$

For example, using Google we get:

$$\frac{C(\textit{it is easy to see the})}{C(\textit{it is easy to see})} = \frac{54,100,000}{127,000,000} = 0.426$$

# Language Models

Where to get large collections of text (i.e. 'corpora'):

- Linguistic Data Consortium
- European Language Resources Association
- Academic Torrents
- Guttenberg Project
- University of Oxford Text Archive
- Stanford list
- ...

# *Language Models*

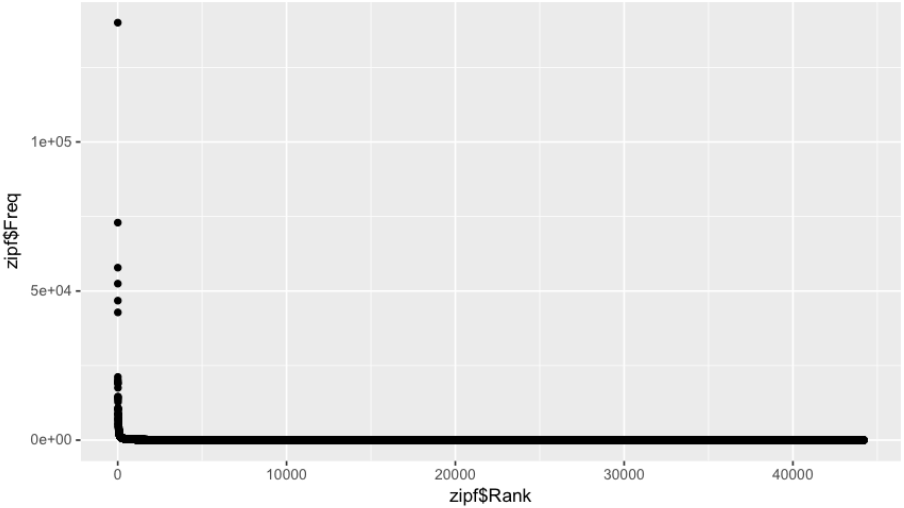
**Problem**: some perfectly normal strings don't occur very much:
$C($*it is easy to see the leaf*$) = 0$

**Sparseness**: most word sequences will never be observed simply because there are too many possibilities. Perfectly grammatical word sequences and their respective Google counts:

(1) a. "going to Tonawanda" (4,370)
b. "going to Dayton" (221K)
c. "going to NYC" (1M)
d. "John is not showering" (2)
e. "John is not singing" (5,010)

# Language Models

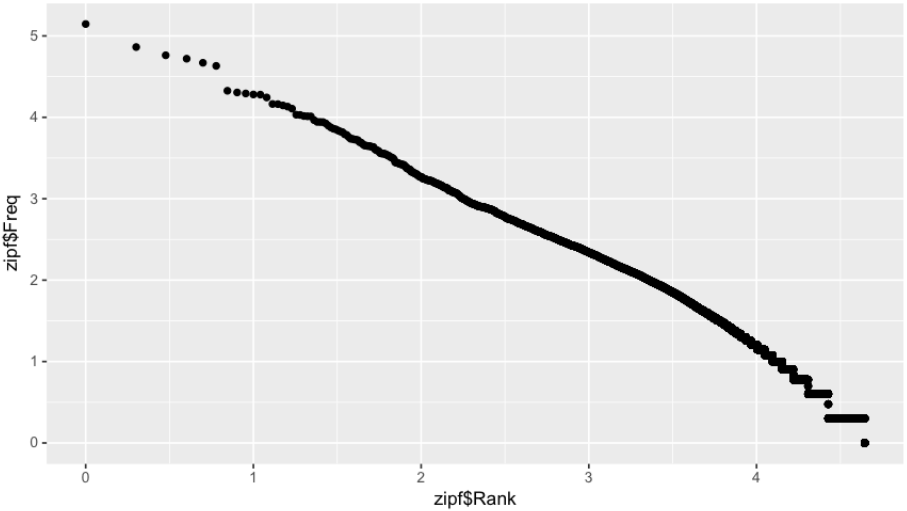
Zipf's long tail distribution:




(Data extracted from the Brown corpus)

# Language Models

Zipf's long tail distribution:




(Data extracted from the Brown corpus)

**Zipf's long tail distribution**:

- A small number of events have high frequency (easy to collect)
- A large number of events have low frequency (hard to collect)

**Zipf's Law**:
the probability of the n-th most frequent word $w$ is roughly

$$P(w) = \frac{0.1}{n}$$

In any book, about 50% of words are the same 50 to100 words, while the other 50% of words appear only once (e.g. Alice in Wonderland 56% appear only once, Tom Sawyer 50.2%, etc.)

Incidentally, similar power laws appear in:

- Character frequency
- Week day mention frequency
- Protagonist mention frequency
- City populations
- Solar flare intensities
- Protein sequences
- Website traffic
- Earthquake magnitudes
- Citation frequency
- Last names
- Diameter of Moon craters
- Phone call frequency
- Opening chess move frequency
- Rate of memory decay

Related to the Pareto distribution

# Language Models

**Pareto Principle**: 20% of the causes are responsible for 80% of the outcome.

- 20% of humans have 82.7% of the world's income
- 20% of patients use 80% of healthcare resources
- 20% of the bugs detected cause 80% of crashes in Microsoft
- 20% of the carpet in an office receives 80% of the ware
- ...

E.g. there are a few things that matter a lot, and much that doesnt matter at all.

E.g. the more something happens, the more likely it is to happen in the future.

E.g. Matthew effect:

> *For to every one who has will more be given, and he will have abundance; but from him who has not, even what he has will be taken away. – Matthew 25:29.*

# *Language Models*

**Approach #2:** estimate $P($*it is easy to see the*$)$ using n-grams

An n-gram is a sequence of wordforms (word types) and its probability.

- Unigram: a single wordform
- Bigram: a sequence of two wordforms
- Trigram: a sequence of three wordforms
- 4-gram: a sequence of four wordforms
- ... etc.

Online n-gram calculator.
Google's n-gram viewer.
Automatic random language generation

# Language Models

- **Chain Rule of Probability** (general case)

  $P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_1^{k-1}) =$
  $P(w_1) \times P(w_2|w_1) \times P(w_3|w_1^2) \times ... \times P(w_n|w_1^{n-1})$

  It is usually impractical to compute all these probabilities...

- **Markov (simplifying) assumption**: the values of any state are only influenced by the values of the state that directly preceded it.

  Markov estimation of the joint probability (bigram version)

  $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$

**Maximum likelihood estimation** (using relative frequency)

In the bigram case:

$$P_{MLE}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

The count $C(w_{n-1}w_n)$ is normalized by dividing by the sum of all the bigrams that start with the same word.

But since that is equivalent to a unigram count, we have:

$$P_{MLE}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

In general, for N-grams we have:

$$P_{MLE}(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

## Example

**Bigrams** (0 cells do not need to be explicitly listed)

| | **i** | **want** | **to** | **eat** | **chinese** | **food** | **lunch** | **spend** |
|---|---|---|---|---|---|---|---|---|
| **i** | 5 | 827 | 0 | 9 | 0 | 0 | 0 | 2 |
| **want** | 2 | 0 | 608 | 1 | 6 | 6 | 5 | 1 |
| **to** | 2 | 0 | 4 | 686 | 2 | 0 | 6 | 211 |
| **eat** | 0 | 0 | 2 | 0 | 16 | 2 | 42 | 0 |
| **chinese** | 1 | 0 | 0 | 0 | 0 | 82 | 1 | 0 |
| **food** | 15 | 0 | 15 | 0 | 1 | 4 | 0 | 0 |
| **lunch** | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **spend** | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**Unigram counts:**

| **i** | **want** | **to** | **eat** | **chinese** | **food** | **lunch** | **spend** |
|---|---|---|---|---|---|---|---|
| 2533 | 927 | 2417 | 746 | 158 | 1093 | 341 | 278 |

**Applying LME**: divide each row by the unigram counts.

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| **i** | 0.002 | 0.33 | 0 | 0.0036 | 0 | 0 | 0 | 0.00079 |
| **want** | 0.0022 | 0 | 0.66 | 0.0011 | 0.0065 | 0.0065 | 0.0054 | 0.0011 |
| **to** | 0.00083 | 0 | 0.0017 | 0.28 | 0.00083 | 0 | 0.0025 | 0.087 |
| **eat** | 0 | 0 | 0.0027 | 0 | 0.021 | 0.0027 | 0.056 | 0 |
| **chinese** | 0.0063 | 0 | 0 | 0 | 0 | 0.52 | 0.0063 | 0 |
| **food** | 0.014 | 0 | 0.014 | 0 | 0.00092 | 0.0037 | 0 | 0 |
| **lunch** | 0.0059 | 0 | 0 | 0 | 0 | 0.0029 | 0 | 0 |
| **spend** | 0.0036 | 0 | 0.0036 | 0 | 0 | 0 | 0 | 0 |

$$P(\text{want}|\text{i}) = \frac{C(\textit{i want})}{C(\textit{i})} = \frac{827}{2533} = 0.33$$

$$P(\cdot \text{ I want chinese food } \cdot) = P(\text{i}|\cdot)\times\ P(\text{want}|\text{i})\times$$
$$P(\text{chinese}|\text{want})\times\ P(\text{food}|\text{chinese})\times\ P(\cdot|\text{food}) =$$
$$0.25 \times 0.33 \times 0.0065 \times 0.52 \times 0.68 = 0.000189618$$

# *Language Models*

With a large corpus, the calculation of the probability of a large sentence can lead to **underflow**: a very small number, close to zero, too long for the computer to handle.

One common solution is to store probabilities in log space:

| | **i** | **want** | **to** | **eat** | **chinese** | **food** | **lunch** | **spend** |
|---|---|---|---|---|---|---|---|---|
| **i** | 0.002 | 0.33 | – | 0.0036 | – | – | – | 0.00079 |

$\text{Log}_{10}$:

| | **i** | **want** | **to** | **eat** | **chinese** | **food** | **lunch** | **spend** |
|---|---|---|---|---|---|---|---|---|
| **i** | -2.65 | -0.48 | – | -2.44 | – | – | – | -3.10 |

Exp:

| | **i** | **want** | **to** | **eat** | **chinese** | **food** | **lunch** | **spend** |
|---|---|---|---|---|---|---|---|---|
| **i** | 0.07 | 0.6 | – | 0.08 | – | – | – | 0.04 |

In which case the probabilities are added rather than multiplied:

$$P_1 \times ... \times P_n = exp(logP_1 + ... + logP_n)$$

# Language Models

Suppose a speech recognition system found two possible tokenizations for a sentence:

(2) a. It's not ice cream

b. It snot I scream

Which sentence is more likely? Below are Log values.

| | **it** | **'s** | **not** | **ice** | **cream** | **snot** | **I** | **scream** |
|---|---|---|---|---|---|---|---|---|
| **it** | -2.34 | -0.27 | – | – | – | – | -3.23 | – |
| **'s** | – | – | -0.12 | -0.89 | -0.73 | -0.68 | – | -0.45 |
| **not** | – | – | – | -0.56 | -0.87 | -0.12 | -0.34 | – |
| **ice** | -0.93 | -0.45 | -1.25 | -4.06 | -0.12 | – | -0.72 | – |
| **cream** | -0.69 | -0.067 | -1.3 | -0.94 | – | -1.9 | -0.84 | – |
| **snot** | -1.58 | -0.89 | -3.43 | – | – | – | -1.34 | – |
| **I** | – | – | – | -3.56 | – | – | – | -0.02 |
| **scream** | -0.05 | -0.38 | -2.04 | – | – | – | -1.08 | – |

Assume that $log(P(\text{it}|\cdot)) = log(P(\cdot|\text{cream}) = log(P(\cdot|\text{scream})) = -0.03$.

$P(\cdot \text{ it's not ice cream}\cdot) = -0.03 - 0.27 - 0.12 - 0.56 - 0.12 - 0.03 = -1.13$

$P(\cdot \text{ it snot I scream}\cdot) = -0.03 + log(0) - 1.34 - 0.02 - 0.12 - 0.03 = \ldots$

# Language Models

Creating a bigram model using basic Linux commands:

```
# find  all single (space-separated) words, using ‘@’
# as a sentence boundary delimiter
egrep -o ’[a-z@]+’ brown.txt > unig1.txt

# create a version of ’unig1.txt’ offset by 1 word
tail -n+2 unig1.txt > unig2.txt

# join each of the lines of the two files
paste unig1.txt unig2.txt  > pairs.txt

# obtain the count of word pair types, sorted by frequency
sort < pairs.txt | uniq -c | sort -n -r > bigrams.txt
```

# Language Models

```
1874 of the
1138 in the
 730 to the
 432 from the
 369 of his
 361 and the
 339 the whale
 335 on the
 333 of a
 324 with the
 321 to be
 320 at the
 310 by the
 294 for the
 253 in his
 246 into the
 245 in a
 234 with a
 220 that the
 216 upon the
 213 the ship
 207 all the
 204 as the
 199 it is
```

# *Language Models*

It is not always practical to use N-grams for large N:

For example, assuming a set of 20,000 word types:

| N-gram | Parameters |
|---|---|
| bigram | $20{,}000^2 = 400$ million |
| trigram | $20{,}000^3 = 8 \times 10^{12}$ |
| four-gram | $20{,}000^4 = 16 \times 10^{16}$ |
| five-gram | $20{,}000^5 = 32 \times 10^{20}$ |

**Stemming** can reduce the parameter size significantly.

# *Language Models*

Let's use Shakespeare's work as a test corpus, for example. We get:

- N=884,647 (tokens)
- V=29,066 (types)
- 300,000 bigrams, out of $V^2 =$ 844 million possible bigrams.
- 99.96% of the possible bigrams have zero frequency
- Some zeros are real zeros: things that can't happen
- Other zeros are just rare events
- Higher order n-grams are worse

# Language Models

**Random language generation**: a sampling approach.

For $0 \leqslant i \leqslant |V|$ (where V is the set of word types; the vocabulary)

$$\sum_{i=0}^{i=|V|} p_i = 1$$

- i = 0
- s = uniform random number between 0 and 1
- while $s \geqslant 0$ do:
  - i = i +1
  - s = s - $p_i$
- return $i$

**For n-grams:**

Select word conditioned on the sentence delimiter symbol, then select second word conditioned on the previous one and so on until you select another sentence delimiter symbol.

**There is a fundamental problem with the Markov assumption**... ... it's false.

(3) a. The car that John bought his wife exploded.
b. Whatever Robin builds Sam destroys.

(4) a. Which target$_i$ did you say that you tried to shoot $\_{}_i$?
b. This is [something]$_x$ that most cognitive scientists think about $\_{}_x$ but never consider the implications of $\_{}_x$.

(5) a. Nobody said that Sam knows anything about Prolog.
b.*Somebody said that Sam knows anything about Prolog.
c. Someone should say if Sam knows anything about Prolog.

The fact that Markov's assumption is false creates actual errors.

See how Google handles the following.

(6) a. This is the company that the bank bought.
    b. This is the company that the bank you like bought.

# *Language Models*

**N-grams are useful, but too shallow**

- Some gibberish has high probability:

  (7) We're there a dude!
  (supposed to be 'word error dude!')

- Sometimes a string is plausible in multiple ways:
  $P(\text{I}_{PRN}\ \text{saw}_{V}\ \text{with}\ \text{her}_{PRN}\ \text{bikes}_{CN-PL})$
  $P(\text{I}_{PRN}\ \text{saw}_{V}\ \text{with}\ \text{her}_{PRN}\ \text{bikes}_{V-3SG})$

  (8) a. The man that I saw with her bikes was a thief.
  b. The man that I saw with her bikes to work every day.

- Sometimes word-knowledge information is needed:

  (9) a. The farmer killed the cow with the axe.
  b. The cow killed the farmer with the axe