Corpus Words

School of Information Studies
Syracuse University

Housekeeping notes

- Find your name in Mini-Talk
  - Contact me if your name is not on the list
- Mini talk: schedule and rubrics (starting from 9/27)– under "content"
- Answer to Lab exercise in Week 1 is available in Content– Week 1
- Two new discussion forums:
  - Final project: looking for teammates;
  - Link to mini-talk.

Housekeeping notes

- Group Project:
  - Two solid criteria:
    - 4-5 members in one group
    - members should come from at least TWO majors
  - Self-enrollment group can be found on “course homepage”– “groups”

Final project

- Example projects from previous semesters
  - Identifying and categorizing offensive language in social media
  - Amazon fine food reviews analysis
  - Topic modeling for U.S. presidential election
  - Sentiment analysis of Trump impeachment 2019
  - Paraphrase generator
  - Sarcasm detection in Twitter
  - Sentiment analysis of Twitter during 2020 presidential election
  - Sentiment analysis of YouTube comments

Outlines

- Lecture
  - Intro to corpus words
  - Corpus statistics
  - Language model
- Lab

What is Corpus Statistics/Linguistics?

- Corpus: a collection of text
- Corpus statistics: a methodology to process text and provide information about the text (**descriptive** purpose)
- Statistical analysis focuses on:
  - Counting word frequencies, referred to as "Unigram Frequencies"
  - Identify patterns of text: concordances, collocations. Collocations of words: bigrams, trigrams, etc.

Preliminary Text Processing Required

- Prepare the data
  - Clean the data
  - Decide what count as a token (punctuation, uppercase vs lowercase)
- Tokenization (or word segmentation):
  - Decide how to separate the characters in the sentence into individual words
  - Requires decisions on how to recognize and deal with punctuation
  - Morphology (To stem or not to stem?)

Word counting– frequency

- Key terms:
  - **Tokens** –words (NLTK treats punctuation as tokens)
  - **Distinct tokens** (or *word types*) distinct words, not counting repetitions
- Goal: count the number of each (distinct) token appearing in the corpus

Word Frequencies

- Count the number of each token appearing in the corpus (or sometimes single document)
- A frequency distribution is a list of all tokens with their frequency, usually sorted in the order of decreasing frequency
- Used to make "word clouds"

university
lab
employer
students
media
information
student
ischool
services
friends
alumni
faculty
syracuse
overview
minor
corporate
collaboratories
academic
ms
job
contact
data
management
school
checklist
career
members
technology
awards
life
centers
center
class
events
visit
science
staff
facilities
campus
resources
advising
cas
labs
board
search
research
advisors
program
graduate
hinds

In-class activity

Please count the number of tokens and distinct tokens in the following sentence.

Note: you need to first define some rules about token/unique token before you count. --- post your answer in Discussion

“Oh, you can’t help that,” said the Cat: “we’re all mad here. I’m mad. You’re mad.”

Corpus Statistics: Unigram/Word Frequencies

School of Information Studies
Syracuse University

How many words in a corpus?

Let N be the number of tokens (words + symbols)

Let V be the size of the vocabulary (the number of distinct tokens)

	Tokens = N	Types = |V|	Lexical richness= V: N
Switchboard phone conversations	2.4 million	20 thousand	≈ 1: 120
Shakespeare	884,000	31 thousand	≈ 1: 28.5
Google N-grams	1 trillion	13 million	≈ 1: 76,900

Zipf's Law

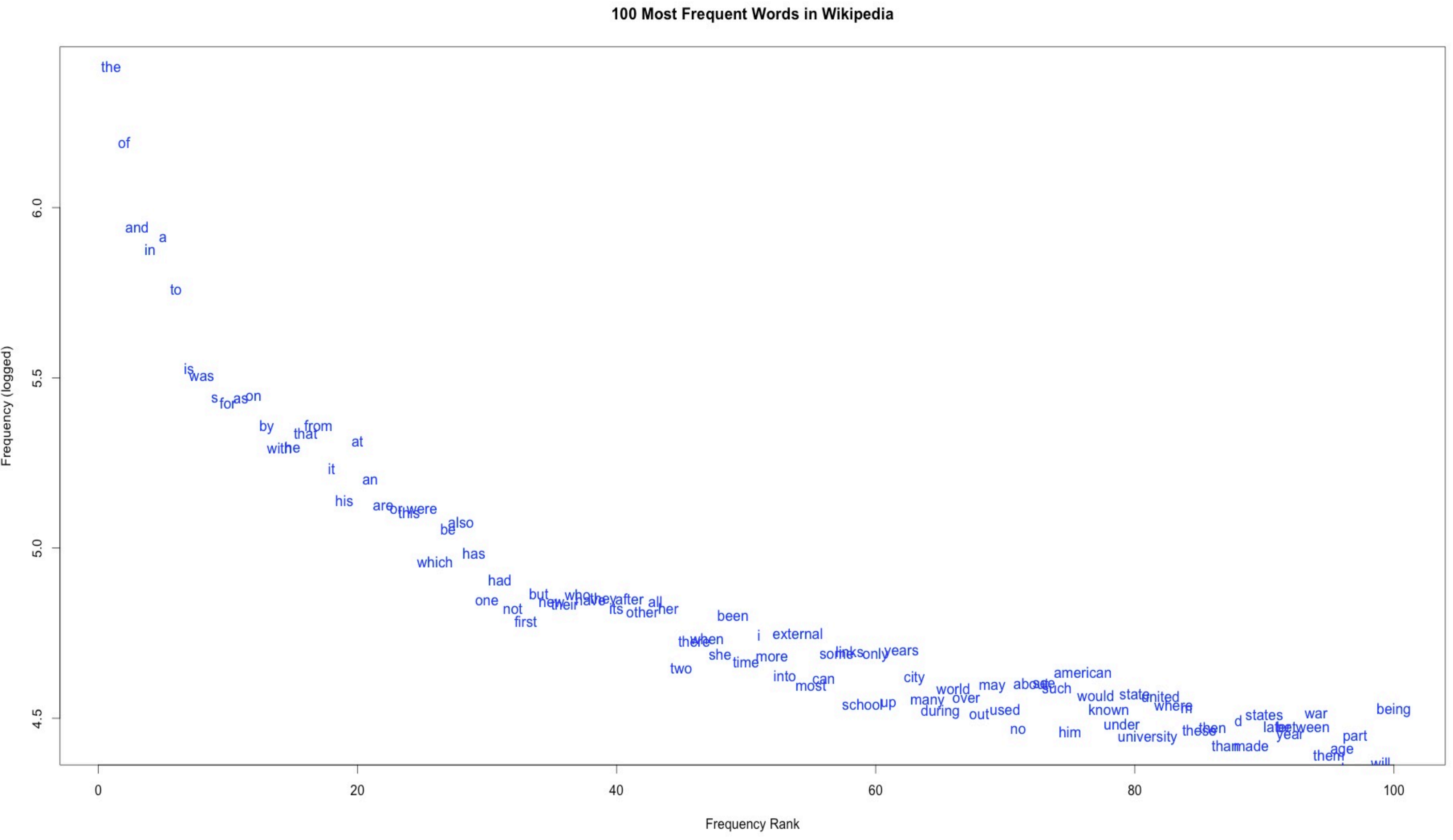
- In a natural language corpus, the frequency of any word is inversely proportional to its rank in a frequency table.
  - The frequency * the rank = constant :

$$\boxed{f \cdot r = k \ \ (\text{for constant } k)}$$

    - r: the numerical position of a word in a list sorted by decreasing frequency (most frequent =1,...)
    - f: the frequency of a word being used in a corpus
    - k: a constant

- Example: if k is 10,000, Zipt's law predicts that:
    - The most frequent word occurs 10,000 times
    - The second most frequent word occurs 5000 times
    - Most frequent word (r = 1) is twice as frequent as 2nd most frequent
    - Most frequent (r = 1) is 3 times as frequent as 3rd most frequent, etc.

For example, in the Brown Corpus of American English text, the word "the" is the most frequently occurring word, and by itself accounts for nearly 7% of all word occurrences (69,971 out of slightly over 1 million). True to Zipf's Law, the second-place word "of" accounts for slightly over 3.5% of words (36,411 occurrences), followed by "and" (28,852).



a sample of 36.8 million words from Wikipedia, over 580,000 word types, nearly half (280,000) occur just once in the sample. --- image and this data from http://wugology.com/zipfs-law/

Zipf's Law Impact on Language Analysis

Good News: Stopwords (commonly occurring words such as "the") will account for a large fraction of text so eliminating them greatly reduces the number of words in a text

Bad News: For most words, gathering sufficient data for meaningful statistical analysis is difficult since they are extremely rare.

Corpus Statistics: Bigram Frequencies & Mutual Information

School of Information Studies
Syracuse University

Bigram & N-gram

- **Bigrams** are any two words that occur together in the text:
  - "two great and powerful groups of nations", the bigrams are "two great", "great and", "and powerful", etc.
- **N-gram**: a sequence of N words
- Examples:
  - Syracuse University (Bigram)
  - Otto the Orange (3-gram or Trigram)
  - Green Lake State Park (4-gram)

N-gram Frequency

- The **frequency of an n-gram** (e.g. bigram) is the **percentage** of times the n-gram occurs in all the n-grams of the corpus and could be useful in corpus statistics
  - For bigram xy frequency:
    - Count of bigram xy / Count of all bigrams in corpus
  - Examples are in the Google N-gram corpus

Google n-gram viewer

In 2010, Google placed on on-line n-gram viewer that would display graphs of n-gram frequencies of one or more n-grams, based on a corpus defined from Google Books

- Dataset was originally generated in 2009, and then updated in 2012, and 2019.
- https://books.google.com/ngrams

Additional corpus measures

- So far, we have looked at one measure involving bigrams (and these definitions can be extended to n-grams):
  - Bigram frequency – **percentage** occurrence of the bigram in the corpus
    - Seen in the Google N-gram data
- Other measures can be defined about the occurrences of bigrams in a corpus
  - Mutual information, …
    - More of these can be found in the NLTK

Corpus Statistics: Mutual Information (MI)

- Mutual Information computes probability of two words occurring in sequence
- Given a pair of words, compares probability that the two occur together as a joint event to the probability they occur individually
  - Their co-occurrences are simply the result of chance
  - The more strongly connected two items are, the higher will be their MI value

Mutual Information

- Based on work of Church & Hanks (1990), generalizing MI from information theory to apply to words in sequence
  - They used terminology *Association Ratio*

- P(x) and P(y) are estimated by the number of observations of x and y in a corpus and normalized by N, the size of the corpus

- P(x,y) is the number of times that x is followed by y in a bigram

- Mutual Information score (also sometimes called PMI, Pointwise Mutual Information):

$$\mathrm{PMI}\ (x,y) = \log_2\ (\ P(x,y)\ /\ P(x)\ P(y)\ )$$

MI values based on 145 WSJ articles

x	freq (x)	y	freq (y)	freq (x,y)	MI
Gaza	3	Strip	3	3	14.42
joint	8	venture	4	4	13.00
Chapter	3	11	14	3	12.20
credit	15	card	11	7	11.44
average	22	yield	7	5	11.06
appeals	4	court	47	4	10.45
…..					
said	444	it	346	76	5.02

Uses of Mutual Information

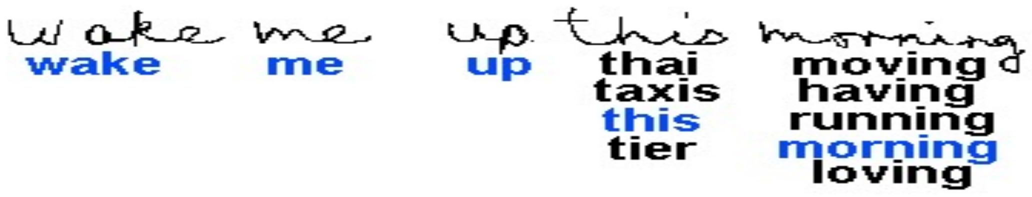
- Can be used to <u>characterize text in Corpus Statistics</u>, but also in other NLP processes
  - Idiomatic phrases for Machine Translation
  - Sense disambiguation
  - Error detection & correction in speech analysis and spell-checking
- Used for distributional semantics in “deep learning”
- Used in applications such as comparing features in machine learning

Language Models/ N-gram Models

School of Information Studies
Syracuse University

Language Models

- The goal of a Language Model is to assign a probability that a sentence (or phrase) will occur in natural uses of the language
- Examples– proper word selection
  - Machine Translation:
    - P(**high** winds tonight) > P(**large** winds tonight)
  - Spell Correction
    - The office is about fifteen **minuets** from my house
      - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - OCR



Corpus Statistics captures a static view of a corpus (**description**), but Language Models use a corpus to **predict** how words occur in sentences.

Language Models

- Goal: compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$

- Related task: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$

  conditional probability that w_5 occurs, given that we know that w_1,w_2,w_3,w_4 already occurred.

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

Chain Rule Applied

- Compute the probability of a sentence by computing the joint probability of all the words conditioned by the previous words

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_1 w_2 \dots w_{i-1})$$

Example: P("its water is so transparent") =
P(its) × P(water|its) × P(is|its water)
× P(so|its water is) × P(transparent|its water is so)

But there are way too many unique English sentences in any realistic corpus for this to work! We'll never see enough data.

Markov Assumption

Andrei Markov

So Instead:

We make the simplifying Markov assumption that we can predict the next word based on only one word previous:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

Or perhaps two words previous:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

N-gram models

Unigram Model: (word frequencies)
The simplest case is that we predict a sentence probability just based on the probabilities of the words with no preceding words

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

Bigram Model: (two-word frequencies)
Prediction based on one previous word:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

N-gram Models

- We can extend to trigrams, 4-grams, 5-grams
  - Each higher number will get a more accurate model, but will be harder to find examples of the longer word sequences in the corpus
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:

    "The computer which I had just put into the machine room on the fifth floor crashed."

  - the last word *crashed* is not very likely to follow the word *floor*, but it is likely to be the main verb of the word *computer*

N-gram predictive probabilities

- For N-grams, we need the conditional probability:

  P(<next word> | <preceding word sequence of length n>)

  e.g. P (*the* | *They picnicked by*)

- We define this as
  - the observed frequency (count) of the whole sequence divided by
  - the observed frequency of the preceding, or initial, sequence
  - sometimes called the maximum likelihood estimation (MLE):

P(<next word> | <preceding word sequence of length n>)

= Count (<preceding word sequence> <next word>)

/ Count (<preceding word sequence>)

  - Example: Count (They picnicked by the) / Count (They picnicked by)

Language Modeling Examples

School of Information Studies
Syracuse University

Example of Bigram predictive probabilities

- Divide the count of the bigram by the count of the first word:

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

- Example: mini-corpus of three sentences, where we have separated sentences, and we include the sentence tags in order to represent the beginning and end of the sentence.

  <S> I am Sam </S>
  <S> Sam I am </S>
  <S> I do not like green eggs and ham </S>

- Bigram predictive probabilities:

  P (I | <S>) = 2/3 = .67 (probability that I follows <S>)
  P (</S> | Sam) = ½ = .5
  P (Sam | <S>) = 1/3 = .33
  P (Sam | am) = ½ = .5
  P (am | I) = 2/3 = .67

In-Class activity– *Go to Kahoot.it*

In the following mini corpus--- **case insensitive**

- <S>Some are old and some are new</S>
- <S>Some are sad, and some are glad</S>
- <S>And some are very, very bad</S>
- <S>Why are they sad and glad and bad</S>

Calculate the bigram predictive probabilities:

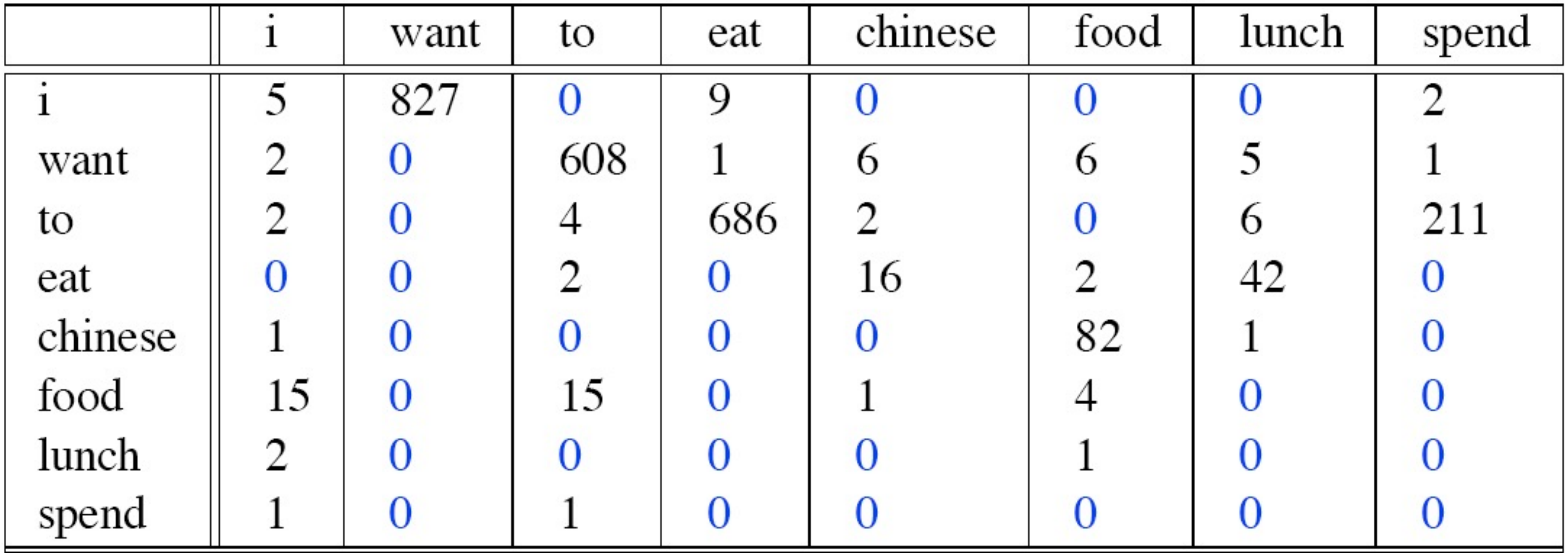
- P(are | some)
- P(some | and)

Example: using bigram predictive probabilities to predict the probabilities of sentences:

- Berkeley Restaurant Project collected online questions from people about restaurants in the Berkeley area.
- Some example sentences
  - can you tell me about any good cantonese restaurants close by
  - mid priced thai food is what i'm looking for
  - tell me about chez panisse
  - can you give me a listing of the kinds of food that are available
  - i'm looking for a good place to eat breakfast
  - when is caffe venezia open during the day

Raw Bigram Counts from the corpus

Out of 9222 sentences, showing counts that the word on the left is followed by the word on the top



	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

Bigram predictive probabilities

Unigram counts:

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

Resulting bigram predictive probability
(bigram counts / unigram counts of first words):

	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

More Bigrams from the restaurant corpus

Eat on	.16	Eat Thai	.03
Eat some	.06	Eat breakfast	.03
Eat lunch	.06	Eat in	.02
Eat dinner	.05	Eat Chinese	.02
Eat at	.04	Eat Mexican	.02
Eat a	.04	Eat tomorrow	.01
Eat Indian	.04	Eat dessert	.007
Eat today	.03	Eat British	.001

Examples due to Rada Mihalcea

Additional Bigrams

<S> I	.25	Want some	.04
<S> I'd	.06	Want Thai	.01
<S> Tell	.04	To eat	.26
<S> I'm	.02	To have	.14
I want	.32	To spend	.09
I would	.29	To be	.02
I don't	.08	British food	.60
I have	.04	British restaurant	.15
Want to	.65	British cuisine	.01
Want a	.05	British lunch	.01

Using N-Grams for sentences

- For a bigram grammar $$\prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

  - P(sentence) can be approximated by multiplying all the bigram probabilities in the sequence

- Example of using bigram probabilities to compute the probability of a sentence:

P(I want to eat Chinese food) =
P(I | <S>) P(want | I) P(to | want) P(eat | to)
P(Chinese | eat) P(food | Chinese)

Computing Sentence Probabilities

P(I want to eat British food) = P(I|<S>) P(want|I) P(to|want) P(eat|to) P(British|eat) P(food|British) = .25×.32×.65×.26×.001×.60 = .000080

vs.

P(I want to eat Chinese food) = .00015

Probabilities seem to capture “syntactic" facts, “world knowledge"

- British food is less popular than Chinese food in this area.

Language Modeling Smoothing

School of Information Studies
Syracuse University

Using n-gram probabilities in a language model: Why do we need smoothing?

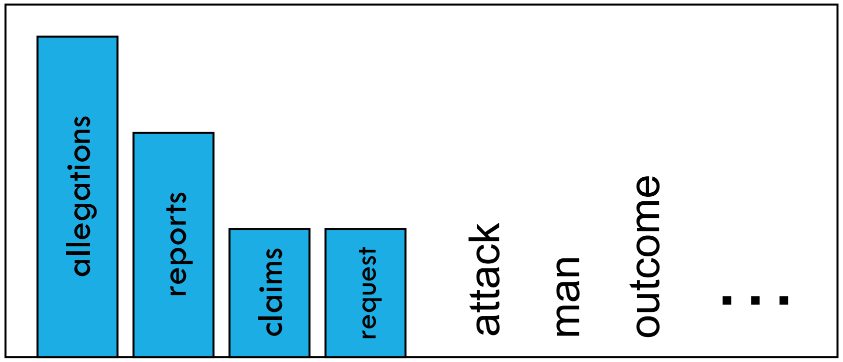
- Every N-gram training matrix is sparse, even for very large corpora (remember Zipf's law)
  - There are words that don't occur in the training corpus that may occur in future text
  - These are known as the unseen words
- Whenever a probability is 0, it will multiply the entire sequence to be 0
- Solution: estimate the likelihood of unseen N-grams and include a small probability for unseen words

Intuition of smoothing (from Dan Klein)

When we have sparse statistics:

P(w | denied the)
- 3 allegations
- 2 reports
- 1 claims
- 1 request

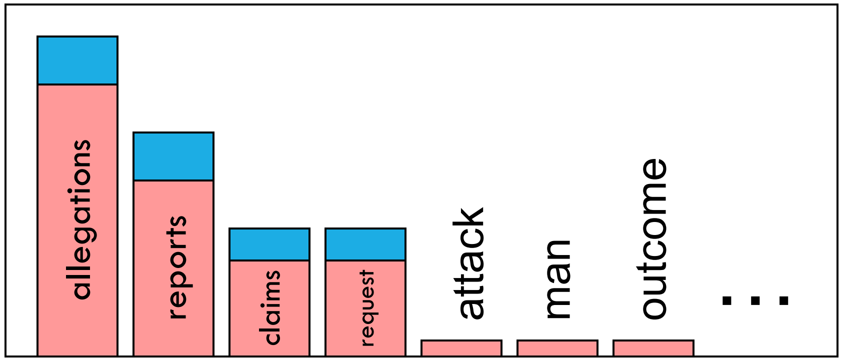
7 total



Steal probability mass to generalize better

P(w | denied the)
- 2.5 allegations
- 1.5 reports
- 0.5 claims
- 0.5 request
- 2 other

7 total



Smoothing

- Add-one smoothing
  - Given: $P(w_n|w_{n-1}) = C(w_{n-1}w_n)/C(w_{n-1})$
  - Add 1 to each count: $P(w_n|w_{n-1}) = [C(w_{n-1}w_n) + 1] / [C(w_{n-1}) + V]$
- Backoff Smoothing for higher-order N-grams
  - Notice that:
    - N-grams are more precise than (N-1)grams
    - But also, N-grams are sparser than (N-1) grams
  - How to combine things?
    - Attempt N-grams and back-off to (N-1) if counts are not available
    - E.g. attempt prediction using 4-grams, and back-off to trigrams (or bigrams, or unigrams) if counts are not available
- More complicated techniques exist: in practice, NLP LM use Knesser-Ney smoothing

Lab 2

School of Information Studies
Syracuse University

Tasks:

- Word frequency distribution
- Self-defined functions
- Bigrams and bigrams frequency distribution

To-do List

- Weekly lab exercise (Lab #2) in Discussion
- Individual assignment #1 (due on 10/3)
- Check out Individual Assignment #1 under "Assignment"– due on 10/3
- Prepare for mini-talk
- Start to think about your final project