Lecture 9: Language models

Zhizheng Wu

Outline

- What is a language model?
- Applications of language models
- N-gram and chain rule
  - Examples for bigram probabilities
- Evaluating language models
- Smoothing

Give a word

The student is watching ________

Probabilistic language model

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

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

- Probability of an upcoming word

$$P(w_n \mid w_1, w_2, w_3, \ldots, w_{n-1})$$

LM applications

- Machine translation

$$P(\text{Students from my class are the best} \mid \text{我班上的学生是最棒的})$$
$$> P(\text{Students from Stanford are the best} \mid \text{我班上的学生是最棒的})$$

- Natural language generation

$$P(\text{best} \mid \text{Students from my class are the}) > P(\text{average} \mid \text{Students from my class are the})$$

- Speech recognition

$$P(\text{Three students}) > P(\text{Tree students})$$

Language models in daily life

Google

the chinese university

The Chinese University of Hong Kong (CUHK)
Public university in Hong Kong

The Chinese University of Hong Kong
University in Shenzhen, China

the chinese university **of hong kong ranking**

the chinese university **of hong kong acceptance rate**

the chinese university **of hong kong press**

the chinese university **of hong kong shenzhen ranking**

the chinese university of hong kong department of mathematics
Department of Mathematics, The Chinese University of Hong Kong · Room 220, The Chinese University of Hong Kong (CUHK), Lady Shaw Building, University Ave, Ma Liu Shui, Hong Kong

the chinese university **of hong kong zip code**

The Chinese University Press

Language models in daily life

Recipients

this is a test email for CSC3160/MDS6002 course

This is a test email on language model applications. I has a typo. can you corret it?

Probability of next word

$$P(\text{best} \mid \text{Students from my class are the}) = \frac{C(\text{Students from my class are the best})}{C(\text{Students from my class are the})}$$

- C(Students from my class are the best) is count of the phrase "*Students from my class are the best*"

Probability of next word

- Smarter way to estimate the probability

$$P(\text{Students from my class are the best})$$
$$= P(\text{best} \mid \text{the})P(\text{the} \mid \text{are})P(\text{are} \mid \text{class})P(\text{class} \mid \text{my})P(\text{my} \mid \text{from})P(\text{from} \mid \text{Students})P(\text{Students})$$

- Chain rule of probability

$$P(w_{1:n}) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1:2}) \ldots P(w_n \mid w_{1:n-1})$$

N-gram

The student is watching______

Unigram: "The"
Bigram: "The student"
Trigram: "The student is"
4-gram: "The student is watching"

Bigram model

- approximates the probability of a word given **all the previous words** by using only the conditional probability of the **preceding word**

$$P(\text{best} \mid \text{Students from my class are the}) \approx P(\text{best} \mid \text{the})$$

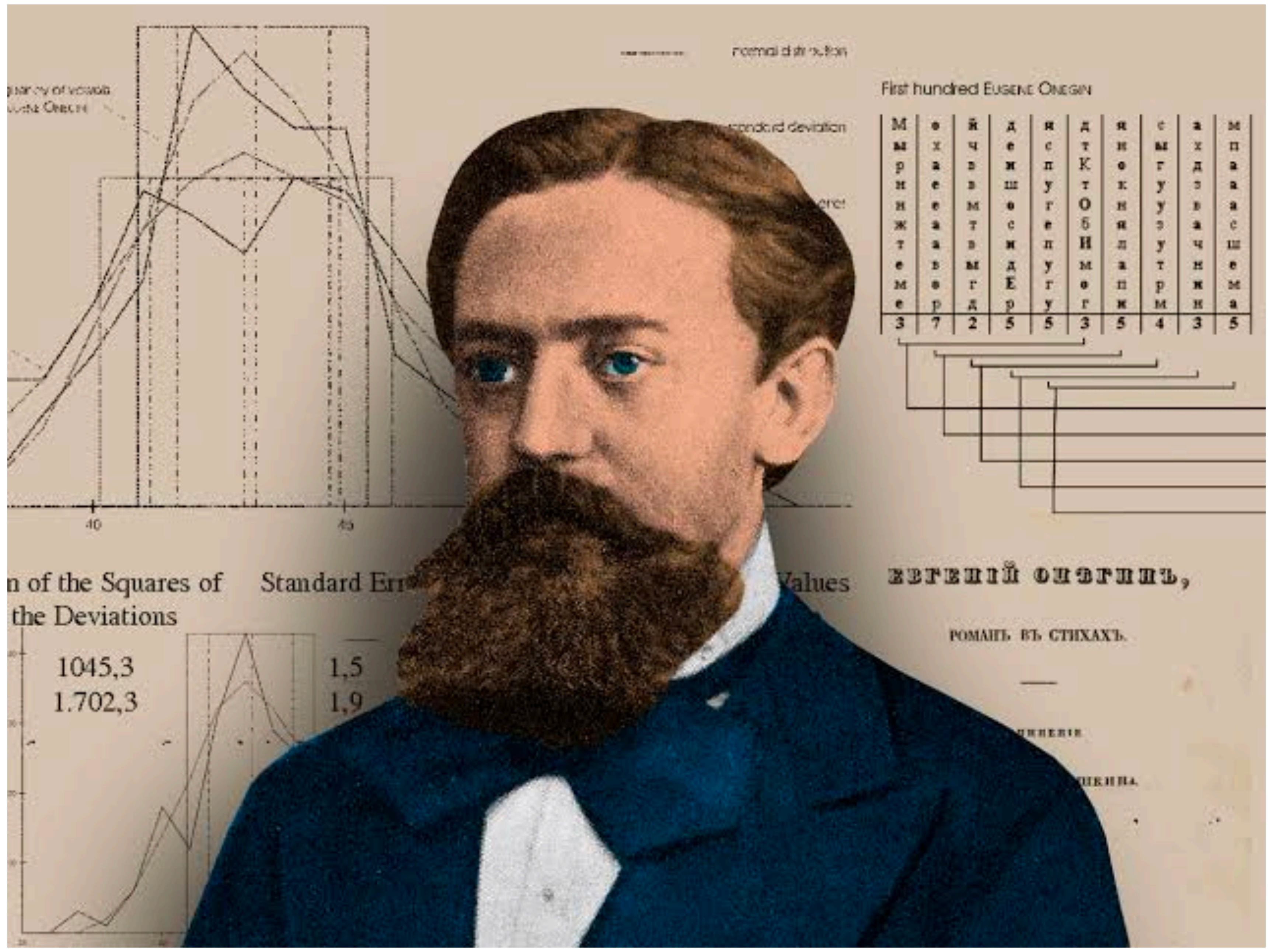
Markov assumption

- Assumption: the probability of a word depends only on the previous word

$$P(\text{best} \mid \text{Students from my class are the}) \approx P(\text{best} \mid \text{the})$$

$$P(w_n \mid w_{1:n-1}) \approx P(w_n \mid w_{n-1})$$

- Markov model: assume we can predict the probability of some future unit without looking too far into the past



Generalizing bigram to n-gram

- From bigram to n-gram

$$P(w_n | w_{1:n-1}) \approx P(w_n | w_{n-N+1:n-1})$$

- N = 2: bigram
- N = 3: trigram
- N = 4: 4-gram
- N = 5: 5-gram

Simplest case: unigram

$$P(w_{1:n}) = P(w_1)P(w_2)P(w_3)\ldots P(w_n)$$

regenerate by simulating an unigram model

Certainly, I can simulate an unigram model to generate a new sentence. Here is an example:

"Dog cat ball blue happy tree sing boy laugh jump sun"

Bigram model

- Condition on the previous word

$$P(w_i \mid w_{1:i-1}) \approx P(w_i \mid w_{i-1})$$

regenerate by simulating a bigram model

Sure, here's an example sentence generated using a bigram model:

"The sun is shining brightly, and the birds are singing in the trees."

Example with a mini-corpus

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

<s> : beginning symbol

</s>: ending symbol

- Maximum-likelihood estimation (MLE): bigram probability

$$P(\texttt{I}|\texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam}|\texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am}|\texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>}|\texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam}|\texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do}|\texttt{I}) = \tfrac{1}{3} = .33$$

$$P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}\ w_n)}{C(w_{n-N+1:n-1})}$$

A slightly large example

- Bigram counts

	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

  - "I want" occurred 827 times in the document.
  - "want want" occurred 0 times.

Bigram probabilities

	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

- Other useful probabilities

$$P(\texttt{i}\,|\,\texttt{<s>}) = 0.25 \qquad P(\texttt{english}\,|\,\texttt{want}) = 0.0011$$
$$P(\texttt{food}\,|\,\texttt{english}) = 0.5 \qquad P(\texttt{</s>}\,|\,\texttt{food}) = 0.68$$

- Calculate probability of sentences like "*I want English food*"

$$\begin{aligned} & P(\texttt{<s> i want english food </s>}) \\ &= P(\texttt{i}\,|\,\texttt{<s>})P(\texttt{want}\,|\,\texttt{i})P(\texttt{english}\,|\,\texttt{want}) \\ &\qquad P(\texttt{food}\,|\,\texttt{english})P(\texttt{</s>}\,|\,\texttt{food}) \\ &= .25 \times .33 \times .0011 \times 0.5 \times 0.68 \\ &= .000031 \end{aligned}$$

Evaluating language models

Perplexity

- the inverse probability of the test set, normalized by the number of words

$$\begin{aligned}\text{perplexity}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}\end{aligned}$$

- Applying chain rule

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

Intuition of perplexity

- Intuitively, perplexity can be understood as a measure of uncertainty

- What's the level of uncertainty to predict the next word?
  - The current president of CUHK Shenzhen is _______ ?
  - ChatGPT is built on top of OpenAI's GPT-3 family of large language _____ ?

- Uncertainty level
  - Unigram: highest
  - Bigram: high
  - 5-gram: low

Lower perplexity = better model

	Unigram	Bigram	Trigram
Perplexity	962	170	109

Model	PPL
Trigram-1	303.2
Trigram-all	112.2
5gram-1	281.0
5-gram-all	73.7
ME-1	286.5
ME-all	68.8
FFNN-all	83.0
RNN-1	211.1
RNN-all	45.7
RNNME-1	196.3
RNNME-3	136.0
RNNME-6	109.7
RNNME-9	107.5
RNNME-12	103.1
RNNME-15	91.3
RNNME-18	106.9
RNNME-21	78.9
L-1-512-512-0.1	63.2
L-1-1024-512-0.1	54.5
L-1-2048-512-0.1	45.3
L-1-8192-2048-0.5	35.9
L-1-8192-2048-0	37.5
L-2-2048-512-0.1	39.8
L-2-4096-1024-0.1	33.6
Human (estimated)	12.0

https://web.stanford.edu/~jurafsky/slp3/3.pdf

https://www.isca-speech.org/archive_v0/Interspeech_2017/pdfs/0729.PDF

Long tail


TO KOYTOYKAKI
TAVERNA
1901
COFFE BAR
CASTELLO PIZZA
ΠΩΛΕΙΤΑΙ
Λαϊκο καφωδειο.
ΤΑΒΕΡΝΑ
Ο ΚΑΝΑΔΟΣ
CASTLE
Imaret
Hotel

The perils of overfitting

- N-gram models only work well for word prediction if the test corpus looks like the training corpus
  - In real world, the inference corpus often doesn't look like the training
  - Robust models that generalize are all we need
  - One kind of generalization: **Zeros**
    - Things that doesn't ever occur in the training set but not in the test set

Zeros

- Training set
  - … denied the allegations
  - … denied the reports
  - … denied the claims
  - … denied the request
- Test set
  - … denied the offer
  - … denied the loan

$$P(\text{offer} \mid \text{denied the}) = 0$$

$$P(\text{loan} \mid \text{denied the}) = 0$$

Zero probability bigrams

- Bigram with zero probability
  - On test set

$$P(w_i \mid w_{1:i-1}) \approx P(w_i \mid w_{i-1})$$

- Perplexity: can’t compute because of 1 over 0…

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

Unseen events

Training data: The wolf is an endangered species
Test data: The wallaby is endangered

Unigram	Bigram	Trigram
P(the)	P(the | <s>)	P(the | <s>)
× P(wallaby)	× P(wallaby | the)	× P(wallaby | the, <s>)
× P(is)	× P(is | wallaby)	× P(is | wallaby, the)
× P(endangered)	× P(endangered | is)	× P(endangered | is, wallaby)

- **Case 1:** P(wallaby), P(wallaby | the), P(wallaby | the, <s>):
  What is the probability of an unknown word (in any context)?
- **Case 2:** P(endangered | is)
  What is the probability of a known word in a known context, if that word hasn't been seen in that context?
- **Case 3:** P(is | wallaby) P(is | wallaby, the) P(endangered | is, wallaby):
  What is the probability of a known word in an unseen context?

https://courses.engr.illinois.edu/cs447/fa2018/Slides/Lecture04.pdf

What can we do?

Dealing with unknown words: Simple solution

- Create an unknown word token <UNK>
  - Training of <UNK> probabilities
  - Create a fixed lexicon **L** of size **V**
  - At text normalization phase, any training word not in L changed to <UNK>

- During inference
  - Use UNK probabilities for any word not in training

Smoothing

- To improve the accuracy of our model
- To handle data sparsity, out of vocabulary words, words that are absent in the training set.
- Smoothing techniques
  - Laplace smoothing: Also known as add-1 smoothing
  - Additive smoothing
  - Good-turing smoothing
  - Kneser-Ney smoothing
  - Katz smoothing
  - Church and Gale Smoothing

Laplace Smoothing

- Assuming every (seen or unseen) event occurred once more than it did in the training data.
-

$$P_{\text{Laplace}}(w_n \mid w_{n-1}) = \frac{C(w_{n-1}, w_n) + 1}{C(w_{n-1}) + V}$$

Bigram counts

Original

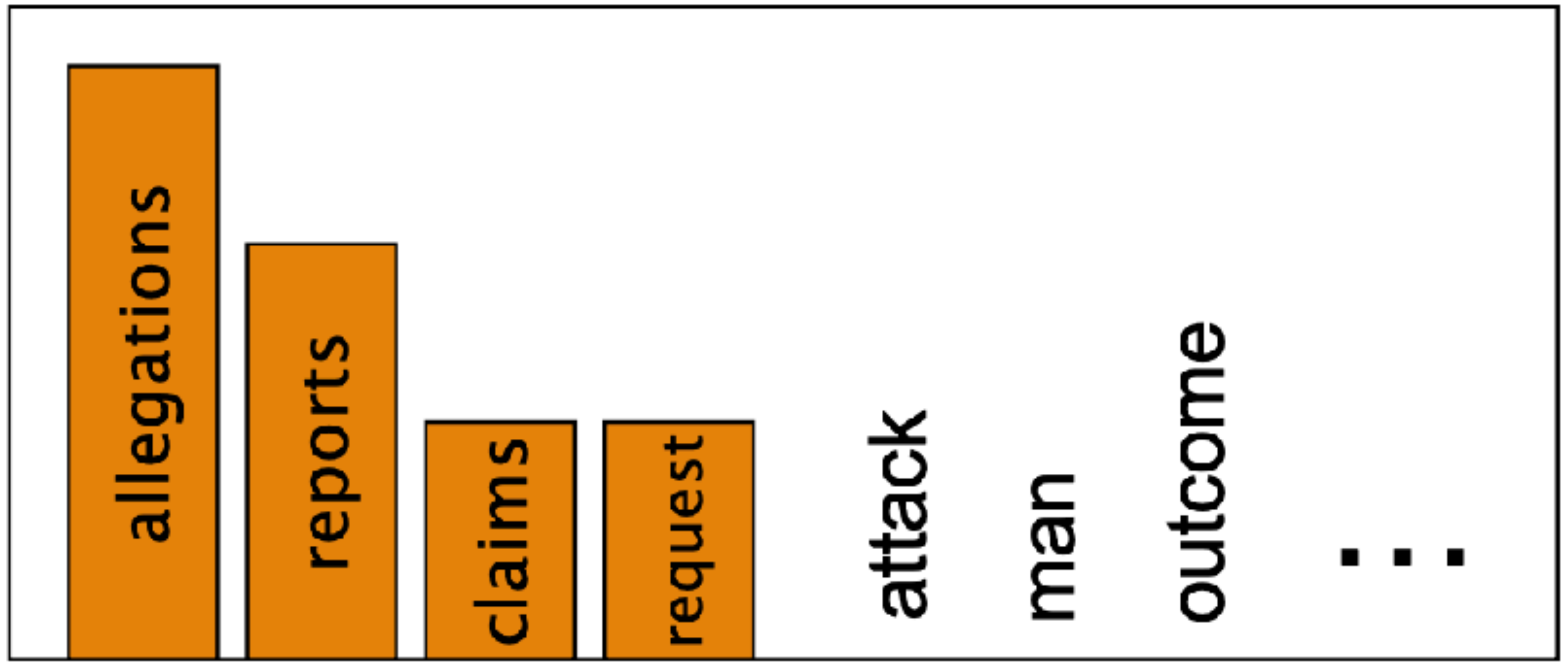
	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

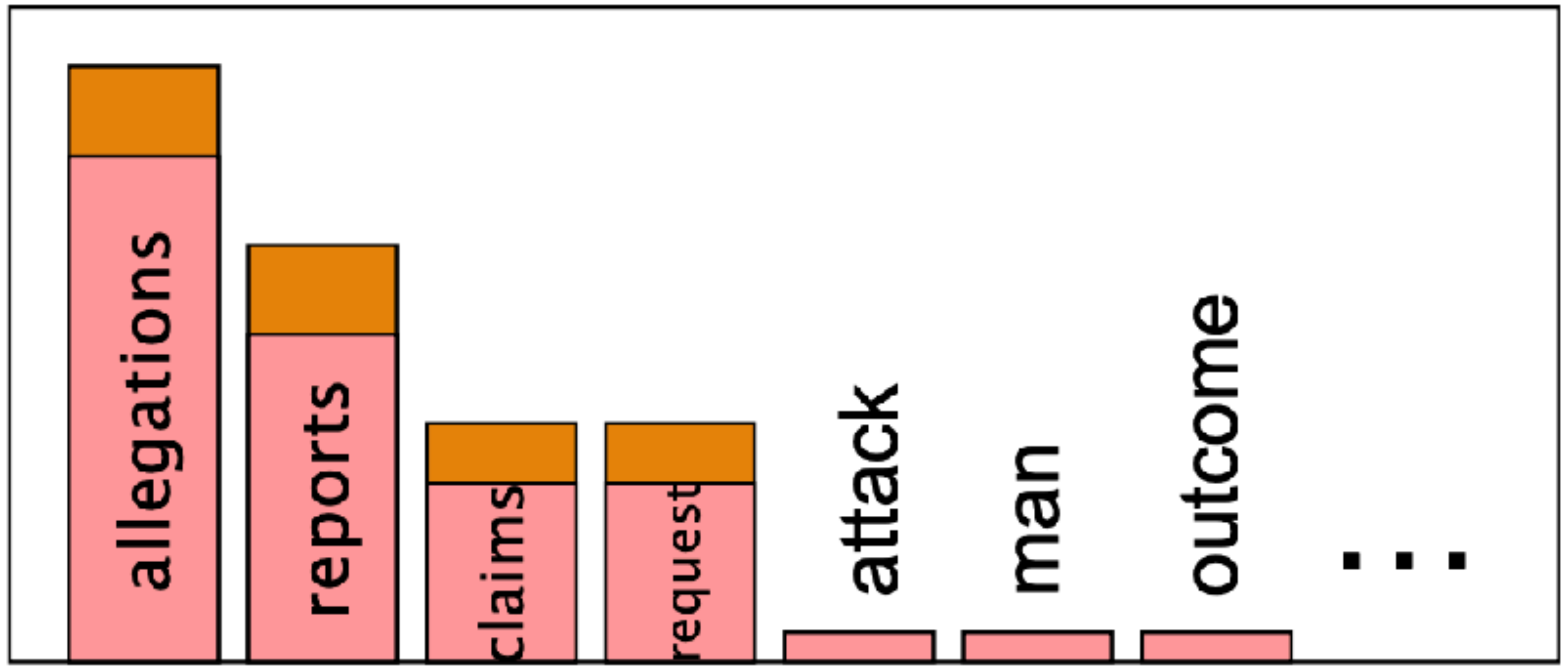
Smoothed

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Intuition of smoothing

- When we have sparse statistics:
  - P(w | denied the)
    - 3 allegations
    - 2 reports
    - 1 claims
    - 1 request
- Steal probability mass to generalize better
  - P(w | denied the)
    - 2.5 allegations
    - 1.5 reports
    - 0.5 claims
    - 0.5 request
    - 2 other





Backoff an interpolation

- Use less context
  - Backoff
    - use trigram if you have good evidence,
    - otherwise bigram, otherwise unigram
  - Interpolation
    - Mix unigram, bigram, trigram

Summary

- Language model
  - Compute the probability of a sentence or sequence of words
  - Predicting next word
- N-gram
  - Unigram
  - Bigram
  - Trigram
  - Etc
- Evaluating language model: perplexity
- Smoothing

Reading

- Chapter 3: N-gram Language Models
  - https://web.stanford.edu/~jurafsky/slp3/3.pdf

Next lecture

- Neural language model
- Large language model