# COS 484: Natural Language Processing

## L2: n-gram Language Models

Spring 2023

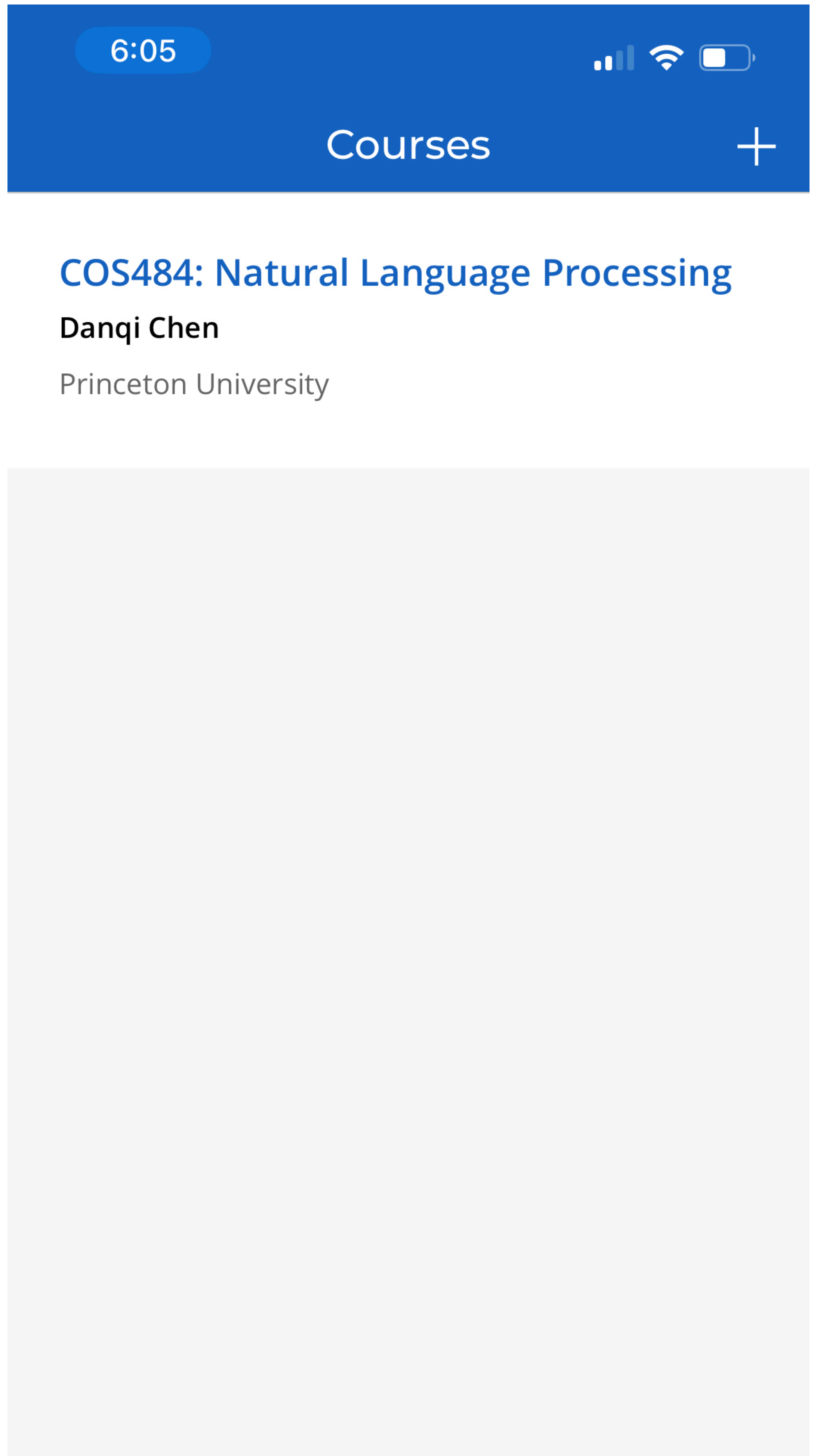
# Announcements

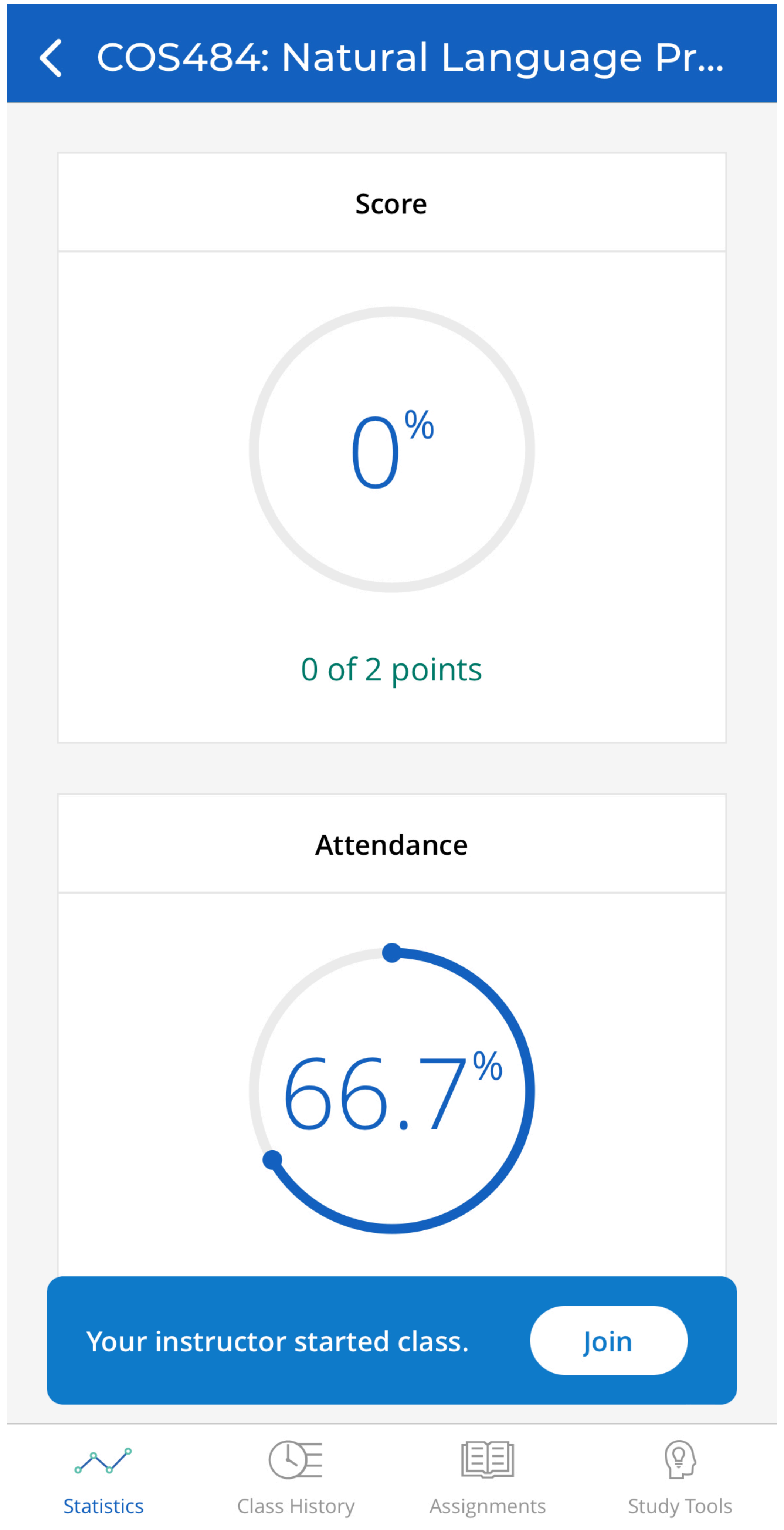
- Office hours posted on the course website (starting from this week)


- Precept time - fill out your availability by Wednesday evening
- Assignment 0 due next Monday at 9:30am

# iClicker setup

- Go to https://join.iclicker.com/ZTYU to join our class
- Open iClicker (e.g., using the iClicker app or going to student.iclicker.com )
- Select our course “COS 484: Natural Language Processing” and click “Join” pop-up box



# Lecture plan

- What is an **n-gram language model**?
- **Generating** from a language model
- **Evaluating** a language model (perplexity)
- **Smoothing**: additive, interpolation, discounting

Most concepts are covered in COS324 already!

**Recommended reading:**
JM3 3.1-3.5

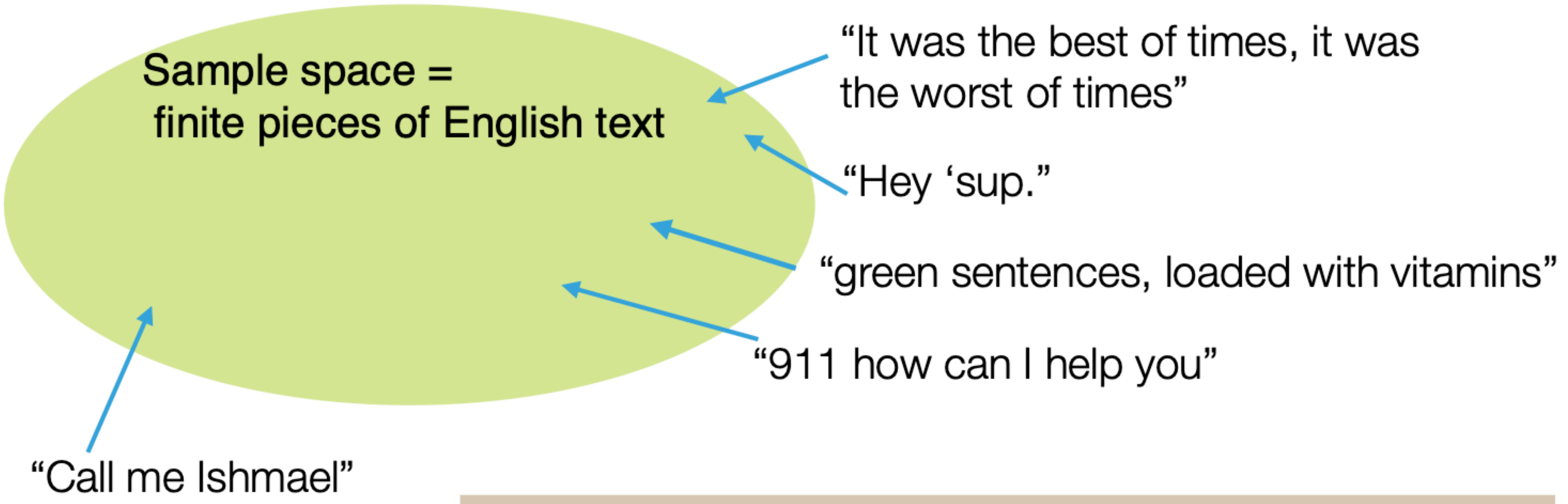
CHAPTER
3
**N-gram Language Models**

# What is an n-gram language model?

# What is a language model?

- A probabilistic model of a sequence of words
- Joint probability distribution of words $w_1, w_2, \ldots, w_n$:

$$P(w_1, w_2, w_3, ..., w_n)$$




(i.e., $\Pr[w_1 w_2 w_2 \ldots w_n]$ associated with every finite word sequence $w_1 w_2 w_2 \ldots w_n$ (including nonsensical ones)

How likely is a given phrase, sentence, paragraph or even a document?

# Chain rule

Conditional probability:
$p(w \mid w_1, w_2), \forall w \in V$

$$p(w_1, w_2, w_3, \ldots, w_n) =$$

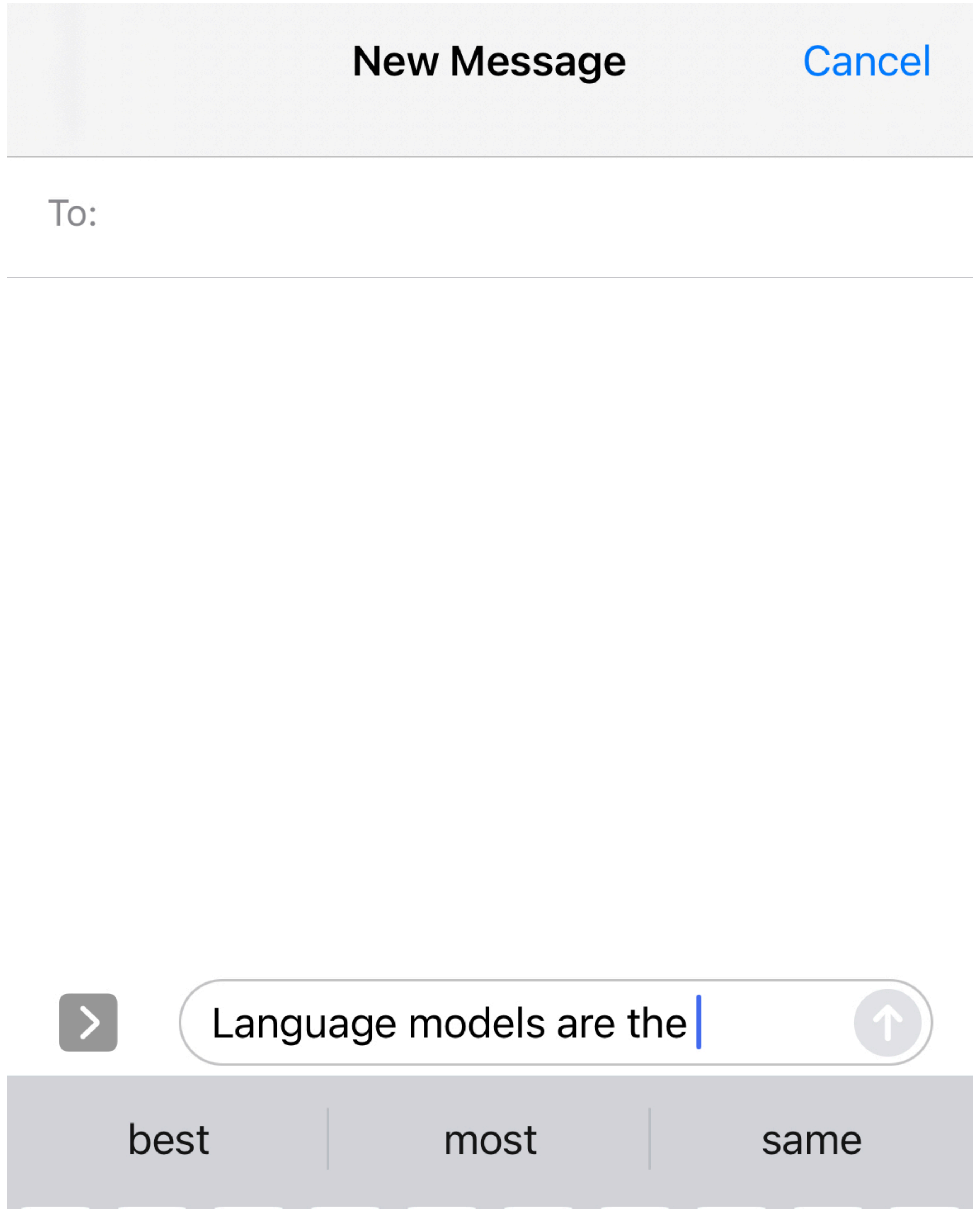
$$p(w_1)p(w_2 \mid w_1)p(w_3 \mid w_1, w_2) \times \cdots \times p(w_n \mid w_1, w_2, \ldots, w_{n-1})$$

Sentence: “the cat sat on the mat”

$$P(\text{the cat sat on the mat}) = P(\text{the}) * P(\text{cat}|\text{the}) * P(\text{sat}|\text{the cat})$$
$$*P(\text{on}|\text{the cat sat}) * P(\text{the}|\text{the cat sat on})$$
$$*P(\text{mat}|\text{the cat sat on the})$$

Implicit order

# Language models are everywhere

# Estimating probabilities

$$P(\text{sat}|\text{the cat}) = \frac{\text{count}(\text{the cat sat})}{\text{count}(\text{the cat})}$$

$$P(\text{on}|\text{the cat sat}) = \frac{\text{count}(\text{the cat sat on})}{\text{count}(\text{the cat sat})}$$

$$\vdots$$

trigram

bigram

Maximum likelihood estimate (MLE)

Assume we have a vocabulary of size $V$,

how many sequences of length $n$ do we have?

A) $n * V$

B) $n^V$

C) $V^n$

D) $V/n$

# Estimating probabilities

$$P(\text{sat}|\text{the cat}) = \frac{\text{count}(\text{the cat sat})}{\text{count}(\text{the cat})}$$

$$P(\text{on}|\text{the cat sat}) = \frac{\text{count}(\text{the cat sat on})}{\text{count}(\text{the cat sat})}$$

$$\vdots$$

Maximum likelihood estimate (MLE)

- With a vocabulary of size V, # sequences of length n = $V^n$
- Typical English vocabulary ~ 40k words
  - Even sentences of length <= 11 results in more than 4 * 10^50 sequences. Too many to count! *(# of atoms in the earth ~ 10^50)*

# Markov assumption

- Use only the recent past to predict the next word
- Reduces the number of estimated parameters in exchange for modeling capacity
- 1st order

$$P(\text{mat}|\text{the cat sat on the}) \approx P(\text{mat}|\text{the})$$

- 2nd order

$$P(\text{mat}|\text{the cat sat on the}) \approx P(\text{mat}|\text{on the})$$

Andrey Markov

# $k^{th}$ order Markov

Consider only the last *k* words (or less) for context

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

which implies the probability of a sequence is:

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

(assume $w_j = \phi \quad \forall j < 0$)

Need to estimate counts for up to (k+1) grams

# n-gram models

Unigram $$P(w_1, w_2, ...w_n) = \prod_{i=1}^{n} P(w_i)$$ e.g. P(the) P(cat) P(sat)

Bigram $$P(w_1, w_2, ...w_n) = \prod_{i=1}^{n} P(w_i | w_{i-1})$$ e.g. P(the) P(cat | the) P(sat | cat)

and Trigram, 4-gram, and so on.

*Larger the n, more accurate and better the language model*
*(but also higher costs)*

*Caveat: Assuming infinite data!*

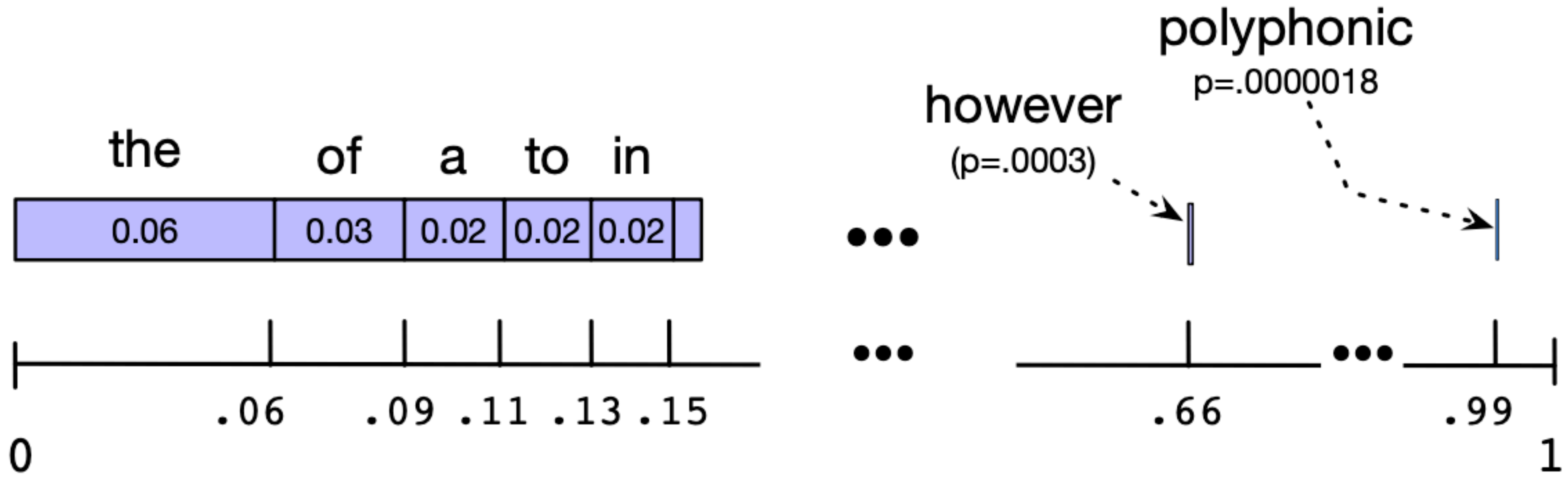
# Generating from a language model

# Generating from a language model

- Given a language model, how to generate a sequence?

Bigram $$P(w_1, w_2, ...w_n) = \prod_{i=1}^{n} P(w_i|w_{i-1})$$

- Generate the first word $w_1 \sim P(w)$
- Generate the second word $w_2 \sim P(w \mid w_1)$
- Generate the third word $w_3 \sim P(w \mid w_2)$
- …



# Generating from a language model

- Given a language model, how to generate a sequence?

Trigram $$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-2}, w_{i-1})$$

- Generate the first word $w_1 \sim P(w)$
- Generate the second word $w_2 \sim P(w \mid w_1)$
- Generate the third word $w_3 \sim P(w \mid w_1, w_2)$
- Generate the fourth word $w_4 \sim P(w \mid w_2, w_3)$
- …

# Generations

**Unigram**

*release millions See ABC accurate President of Donald Will cheat them a CNN megynkelly experience @ these word out- the*

**Bigram**

*Thank you believe that @ ABC news, Mississippi tonight and the false editorial I think the great people Bill Clinton . ''*

**Trigram**

*We are going to MAKE AMERICA GREAT AGAIN! #MakeAmericaGreatAgain https: //t.co/DjkdAzT3WV*

Typical LMs are not sufficient to handle long-range dependencies

"Alice/Bob could not go to work that day because she/he had a doctor's appointment"

# Generations

Example from a GPT-2 output:

prompt aka. conditional generation

**With the start of the new academic year, Princeton** has an opportunity to help provide a new generation of women with a diverse set of academic resources for higher education.
We are offering the resources of the Princeton-McGill program specifically to women with undergraduate degrees who would like to enhance their academic experience. Princeton-McGill offers a comprehensive suite of services for women and their families including a variety of graduate programs, support programs, and the opportunity to serve as leaders in their communities with a wide variety of programs, activities and services. For the upcoming fall, Princeton-McGill will also offer its Women's Center , which is located in a renovated women's dorm.
At Princeton, we are working with the Princeton-McGill community to develop a suite of programs that are designed to give new and returning students a strong foundation for a successful, rewarding graduate career. The Women's Center , the Princeton-McGill Women's Center provides a range of supports to address the specific needs of female doctoral degree graduates. Programs are tailored to meet the unique needs of women under the age of 28, women and families

https://talktotransformer.com/

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1024}, \ldots, w_{i-2}, w_{i-1})$$

Modern LMs can handle much longer contexts!

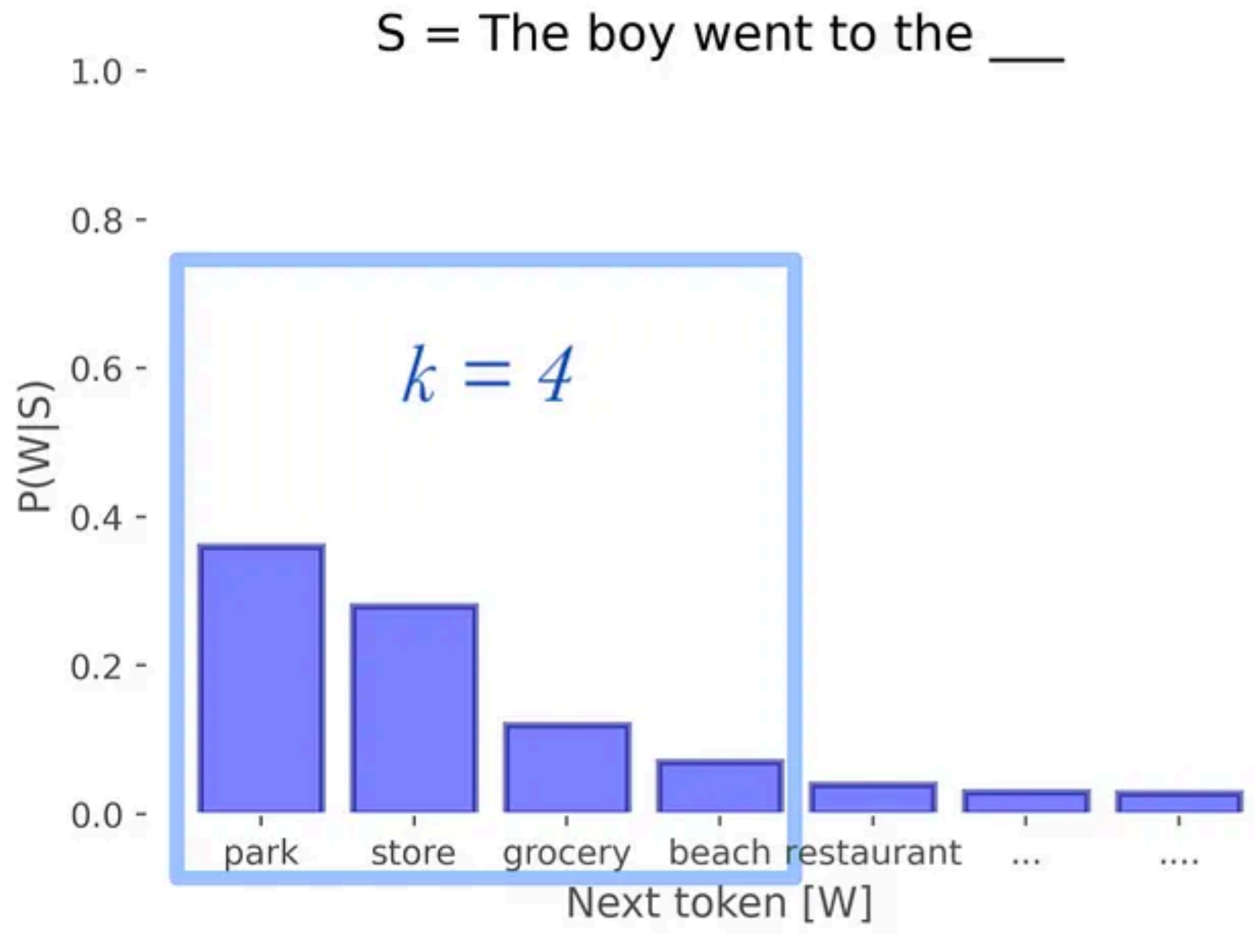
# Generation methods (advanced)

- Greedy: choose the most likely word!

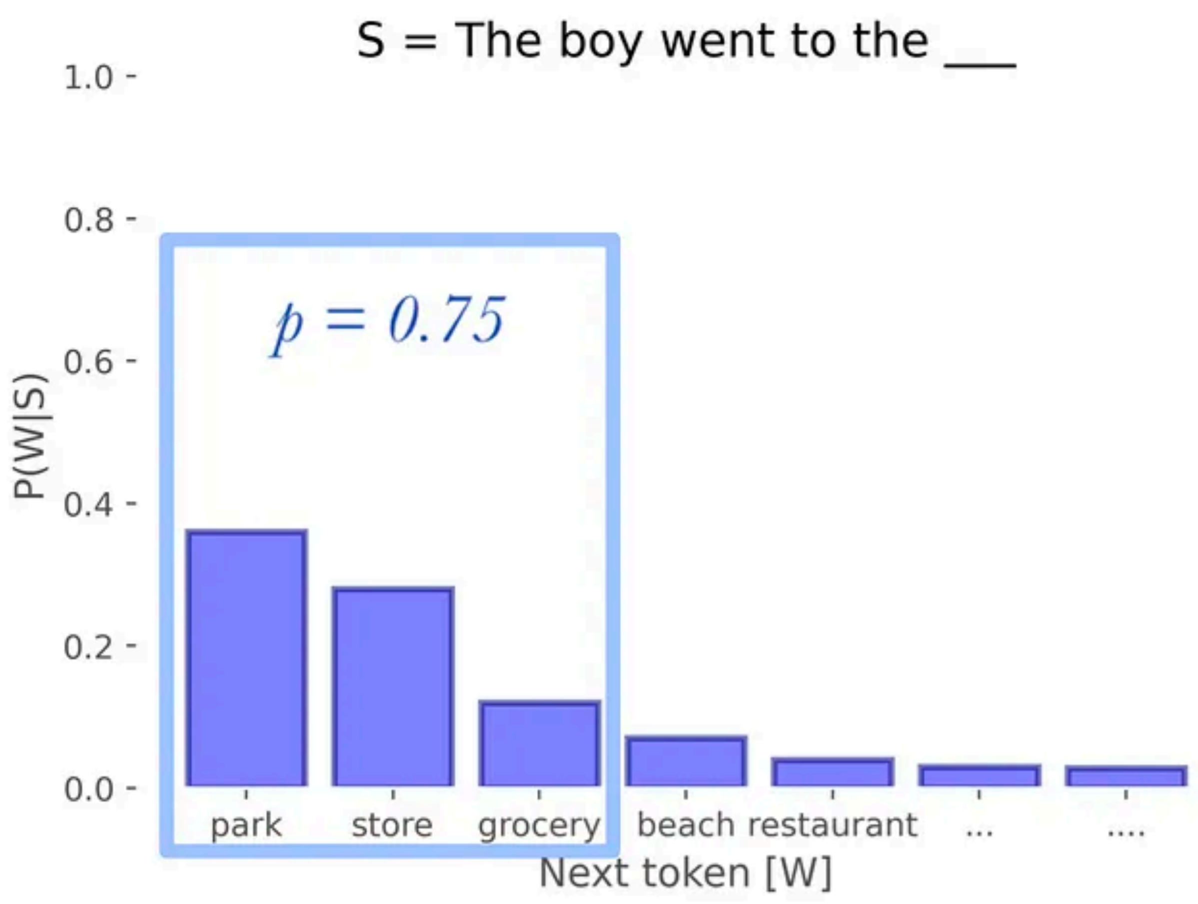
To predict the next word given a context of two words $w_1, w_2$:

$$w_3 = \arg\max_{w \in V} P(w \mid w_1, w_2)$$

- Top-k vs top-p sampling:




Top-k sampling

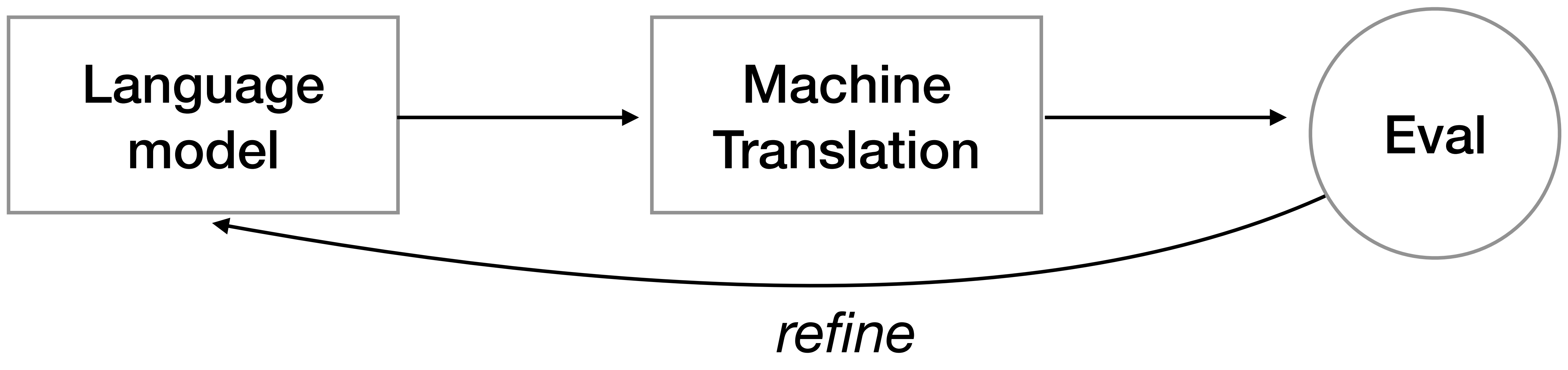



Top-p sampling

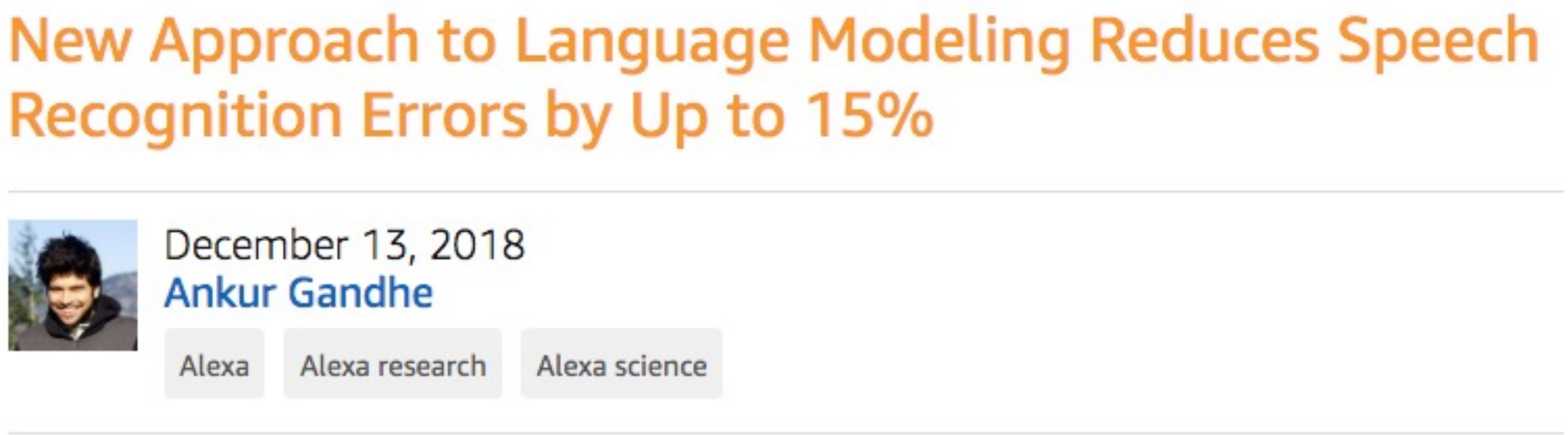
https://blog.allenai.org/a-guide-to-language-model-sampling-in-allennlp-3b1239274bc3

# Evaluating a language model

# Extrinsic evaluation




- Train LM → apply to task → observe accuracy
- Directly optimized for downstream applications
  - higher task accuracy → better model
- Expensive, time consuming
- Hard to optimize downstream objective (indirect feedback)


New Approach to Language Modeling Reduces Speech Recognition Errors by Up to 15%

December 13, 2018
Ankur Gandhe
Alexa
Alexa research
Alexa science

# Intrinsic evaluation of language models

**Research process:**

- Train parameters on a suitable training corpus
  - Assumption: observed sentences ~ good sentences
- Test on *different, unseen* corpus
  - If a language model assigns a higher probability to the test set, it is better
- Evaluation metric - perplexity!

# Perplexity (ppl)

- Measure of how well a LM **predicts** the next word
- For a test corpus with words $w_1, w_2, \ldots w_n$

Perplexity = $P(w_1, w_2, \ldots, w_n)^{-1/n}$

ppl$(S) = e^x$ where $x = -\frac{1}{n} \log P(w_1, \ldots, w_n) = -\frac{1}{n} \sum_{i=1}^{n} \log P(w_i | w_1 \ldots w_{i-1})$

Cross-Entropy

- Unigram model: $x = -\frac{1}{n} \sum_{i=1}^{n} \log P(w_i)$ (since $P(w_j | w_1 \ldots w_{j-1}) \approx P(w_j)$ )

- Minimizing perplexity ~ maximizing probability of corpus

# Intuition on perplexity

If our k-gram model (with vocabulary V) has following probability:

$$P(w \mid w_{i-k}, \ldots w_{i-1}) = \frac{1}{|V|} \quad \forall w \in V$$

what is the perplexity of the test corpus?

A) $e^{|V|}$ B) $|V|$ C) $|V|^2$ D) $e^{-|V|}$

$$\text{ppl}(S) = e^x \quad \text{where} \quad x = -\frac{1}{n} \sum_{i=1}^{n} \log P(w_i \mid w_1 \ldots w_{i-1})$$

Cross-Entropy

# Intuition on perplexity

If our k-gram model (with vocabulary V) has following probability:

$$P(w \mid w_{i-k}, \ldots w_{i-1}) = \frac{1}{|V|} \quad \forall w \in V$$

$$\text{ppl}(S) = e^x \quad \text{where}$$
$$x = -\frac{1}{n}\sum_{i=1}^{n} \log P(w_i \mid w_1 \ldots w_{i-1})$$

what is the perplexity of the test corpus?

A) $e^{|V|}$ B) $|V|$ C) $|V|^2$ D) $e^{-|V|}$

$$\text{ppl} = e^{-\frac{1}{n} n \log(1/|V|)} = |V|$$

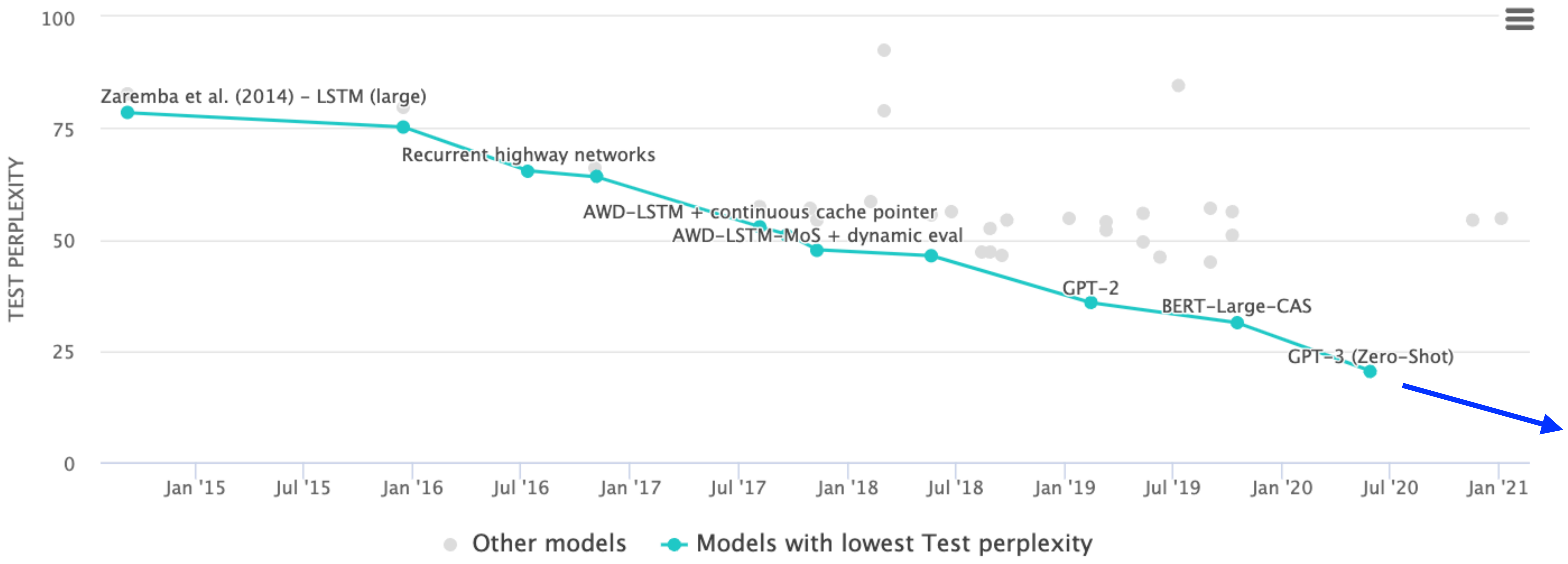
*Measure of model's uncertainty about next word (aka `average branching factor')*

branching factor = # of possible words following any word

# Perplexity

Training corpus 38 million words, test corpus  1.5 million words, both **WSJ**

| N-gram Order | Unigram | Bigram | Trigram |
|---|---|---|---|
| Perplexity (test) | 962 | 170 | 109 |




GPT-3 175B:
ppl = 20.5

https://paperswithcode.com/sota/language-modelling-on-penn-treebank-word

# Smoothing

# Generalization of n-grams

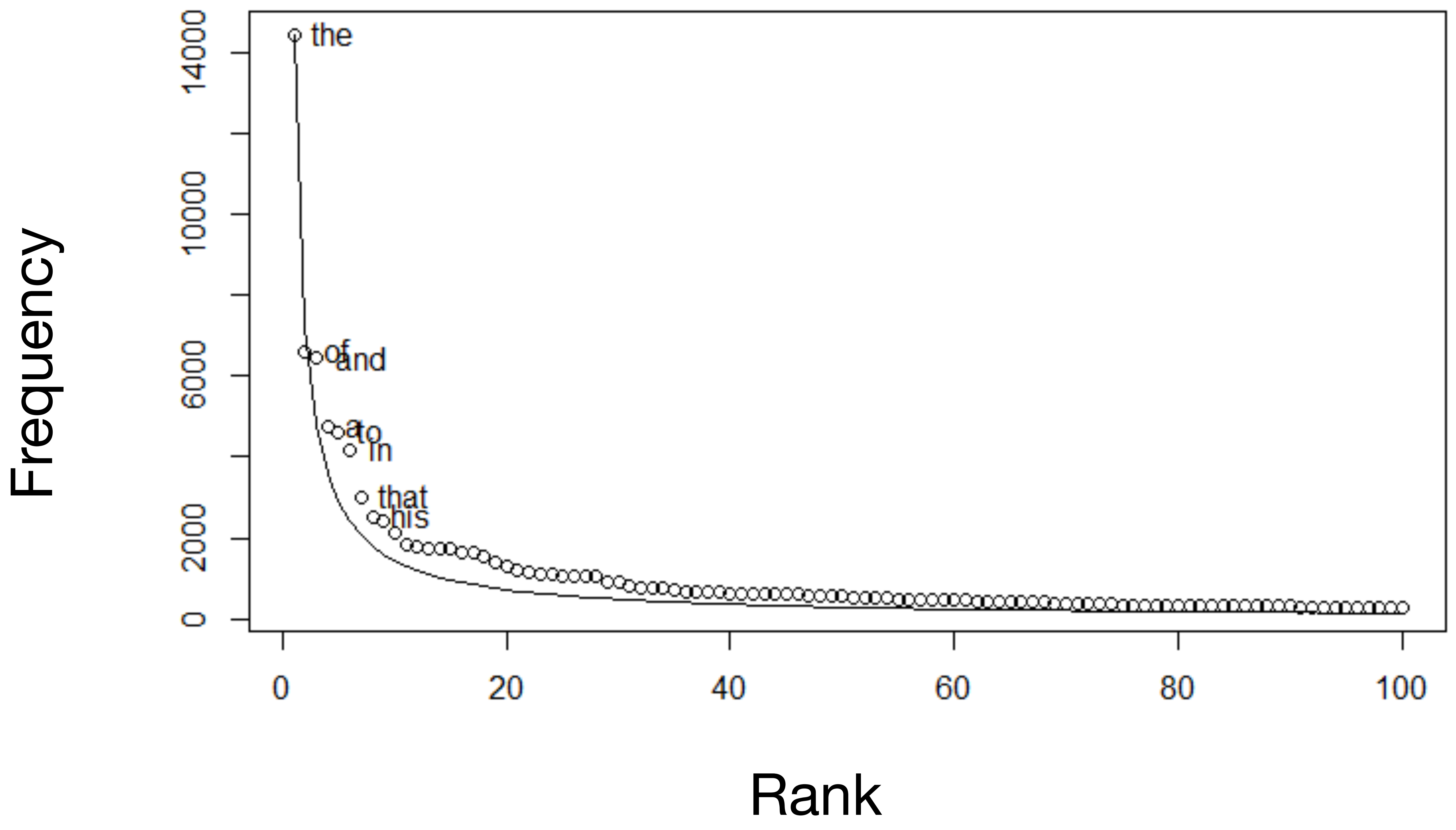
*Any problems with n-gram models and their evaluation?*

- Not all n-grams in the test set will be observed in training data
- Test corpus might have some that have zero probability under our model
  - **Training set**: *Google news*
  - **Test set**: *Shakespeare*
  - P(affray | voice doth us) = 0 $\implies$ P(test corpus) = 0
  - Perplexity is not defined.

$$\text{ppl}(S) = e^{x} \quad \text{where}$$

$$x = -\frac{1}{n}\sum_{i=1}^{n} \log P(w_i \mid w_1 \ldots w_{i-1})$$

# Sparsity in language




$$freq \propto \frac{1}{rank}$$

Zipf's Law

- Long tail of infrequent words
- Most finite-size corpora will have this problem.

# Smoothing

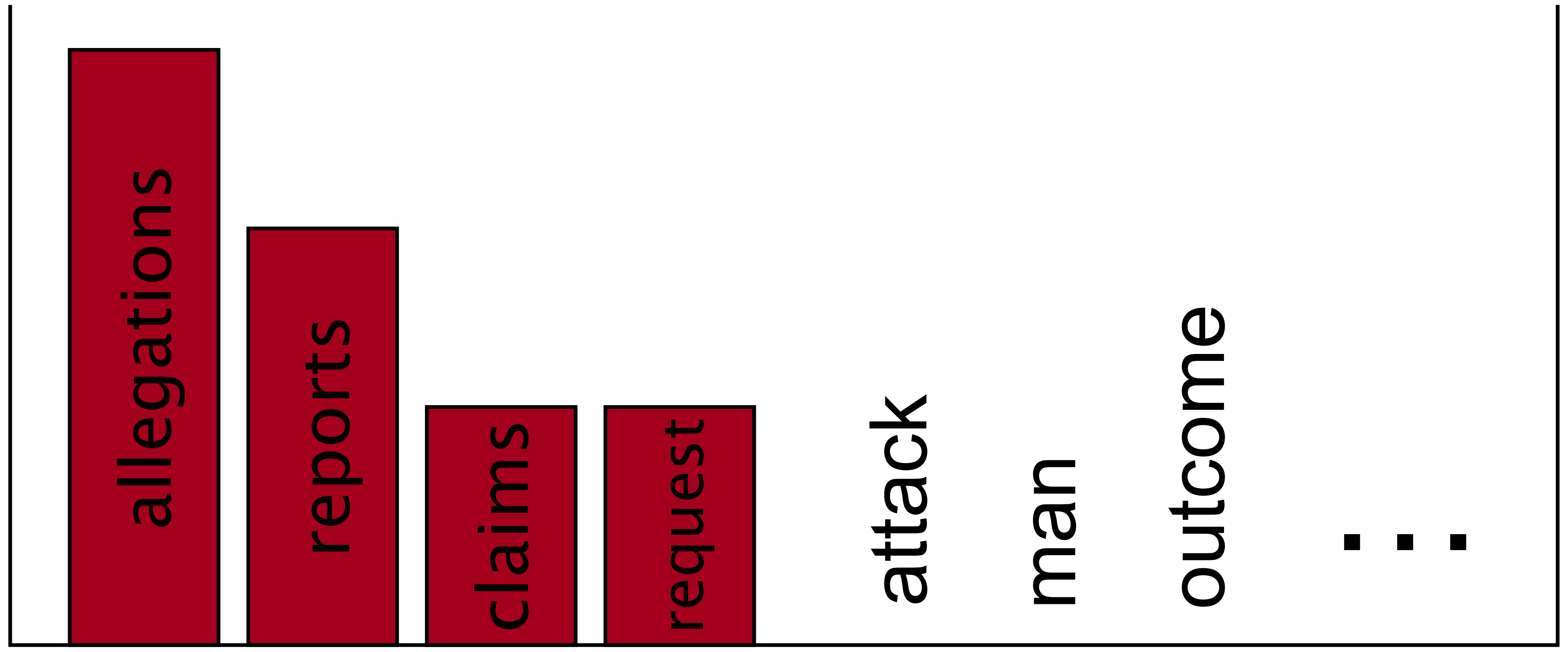
- Handle sparsity by making sure all probabilities are non-zero in our model
    - Additive: Add a small amount to all probabilities
    - Interpolation: Use a combination of different granularities of n-grams
    - Discounting: Redistribute probability mass from observed n-grams to unobserved ones

# Smoothing intuition

When we have sparse statistics:

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

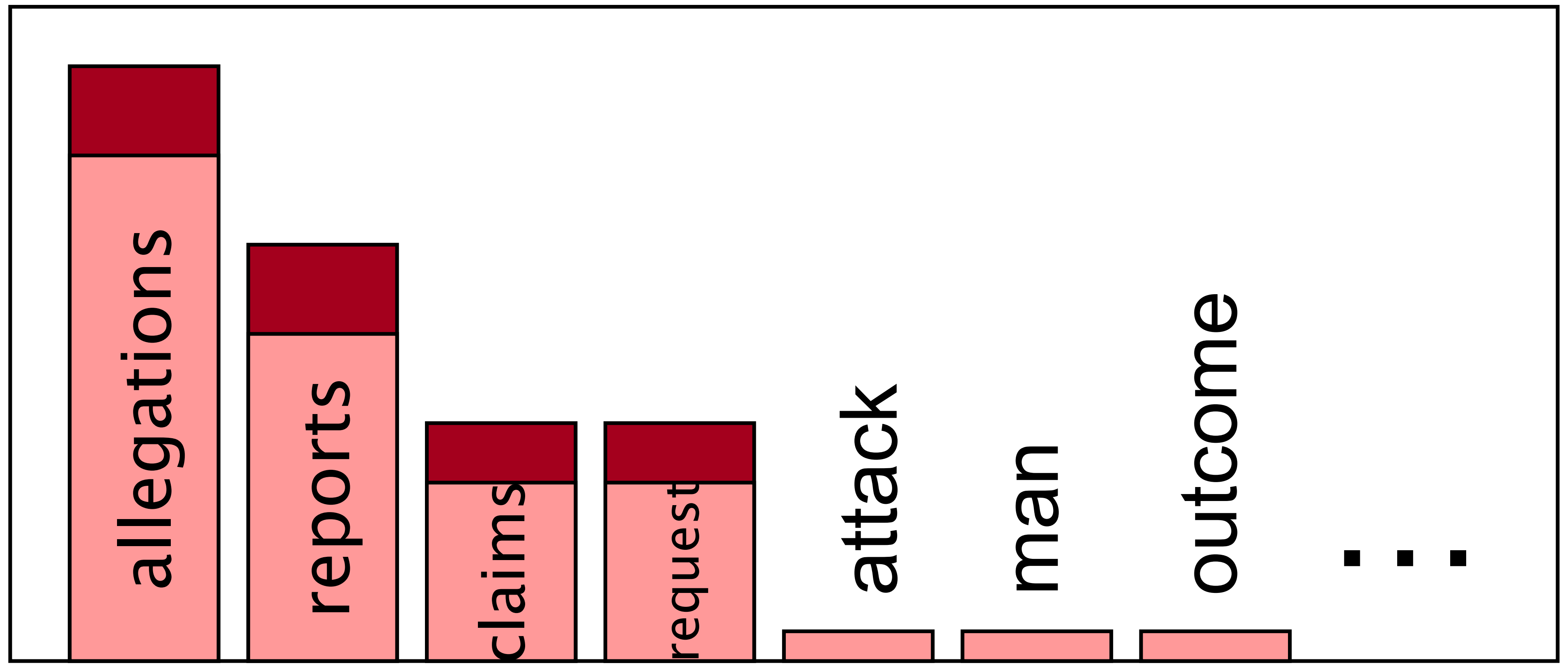
7 total




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




*(Slide credit: Dan Klein)*

# Laplace smoothing

- Also known as add-alpha
- Simplest form of smoothing: Just add $\alpha$ to all counts and renormalize!
- Max likelihood estimate for bigrams:

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

- After smoothing:

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + \alpha}{C(w_{i-1}) + \alpha|V|}$$

# Raw bigram counts (Berkeley restaurant corpus)

- Out of 9222 sentences

| | 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 |

*(Slide credit: Dan Jurafsky)*

# Smoothed bigram counts

| | 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 |

Add 1 to all the entries in the matrix

*(Slide credit: Dan Jurafsky)*

# Smoothed bigram probabilities

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + \alpha}{C(w_{i-1}) + \alpha|V|} \qquad \alpha = 1$$

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 0.0015 | 0.21 | 0.00025 | 0.0025 | 0.00025 | 0.00025 | 0.00025 | 0.00075 |
| want | 0.0013 | 0.00042 | 0.26 | 0.00084 | 0.0029 | 0.0029 | 0.0025 | 0.00084 |
| to | 0.00078 | 0.00026 | 0.0013 | 0.18 | 0.00078 | 0.00026 | 0.0018 | 0.055 |
| eat | 0.00046 | 0.00046 | 0.0014 | 0.00046 | 0.0078 | 0.0014 | 0.02 | 0.00046 |
| chinese | 0.0012 | 0.00062 | 0.00062 | 0.00062 | 0.00062 | 0.052 | 0.0012 | 0.00062 |
| food | 0.0063 | 0.00039 | 0.0063 | 0.00039 | 0.00079 | 0.002 | 0.00039 | 0.00039 |
| lunch | 0.0017 | 0.00056 | 0.00056 | 0.00056 | 0.00056 | 0.0011 | 0.00056 | 0.00056 |
| spend | 0.0012 | 0.00058 | 0.0012 | 0.00058 | 0.00058 | 0.00058 | 0.00058 | 0.00058 |

*(Credits: Dan Jurafsky)*

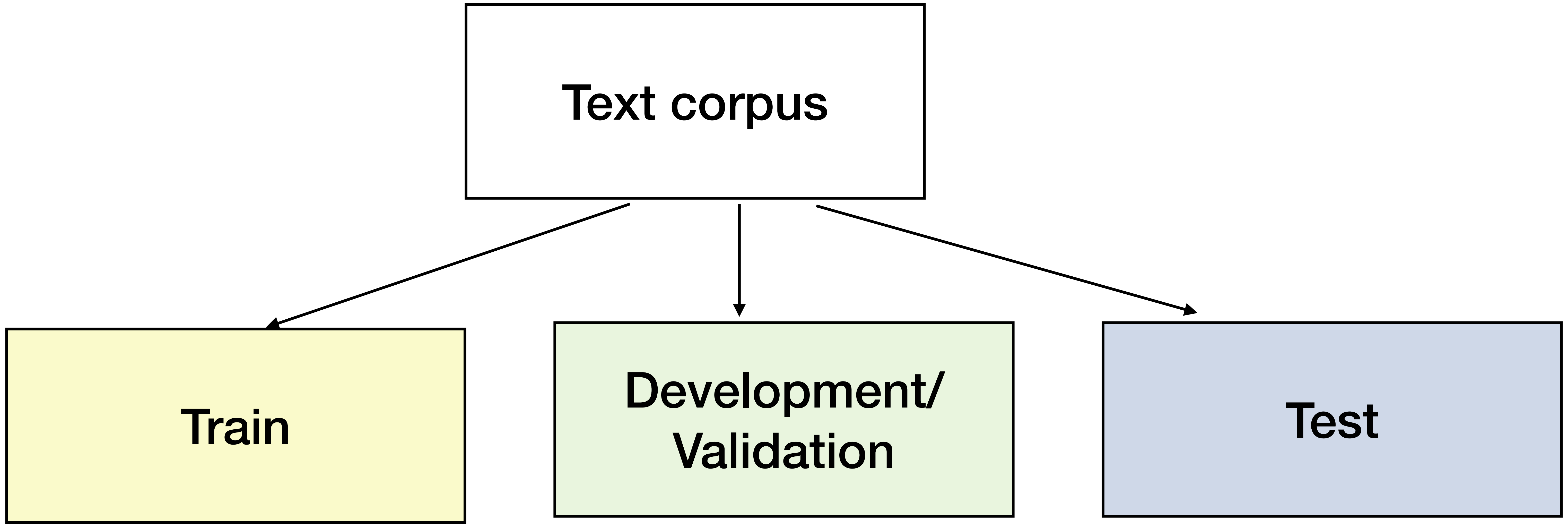
# Linear Interpolation

$$\hat{P}(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 P(w_i \mid w_{i-2}, w_{i-1}) \quad \text{Trigram}$$
$$+\lambda_2 P(w_i|w_{i-1}) \quad \text{Bigram}$$
$$+\lambda_3 P(w_i) \quad \text{Unigram}$$

$$\sum_i \lambda_i = 1$$

- Use a combination of models to estimate probability
- Strong empirical performance

# How can we choose lambdas?




- First, estimate n-gram prob. on training set
- Then, estimate lambdas (hyperparameters) to maximize probability on the held-out development/validation set
- Use best model from above to evaluate on test set

# Average-count (Chen and Goodman, 1996) (advanced)

$= w_{i-n+1}, w_{i-2}, \ldots, w_{i-1}$

$$p_{\text{interp}}(w_i | w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} \, p_{\text{ML}}(w_i | w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) \, p_{\text{interp}}(w_i | w_{i-n+2}^{i-1})$$

(recursive definition)

- Like simple interpolation, but with context-specific lambdas, $\lambda_{w_{i-n+1}^{i-1}}$
- Partition $\lambda_{w_{i-n+1}^{i-1}}$ according to average number of counts per non-zero element:

$$\frac{c(w_{i-n+1}^{i-1})}{|w_i : c(w_{i-n+1}^{i}) > 0|}$$

- Larger $\lambda_{w_{i-n+1}^{i-1}}$ for contexts that appear more often.

# Discounting

- Determine some “mass” to remove from probability estimates
- More explicit method for redistributing mass among unseen n-grams
- Just choose an absolute value to discount (usually <1)

# Absolute Discounting

- Define Count*(x) = Count(x) - 0.5
- Missing probability mass:

$$\alpha(w_{i-1}) = 1 - \sum_{w} \frac{\text{Count}^*(w_{i-1,w})}{\text{Count}(w_{i-1})}$$

$$\alpha(\text{the}) = 10 \times 0.5/48 = 5/48$$

- Divide this mass between words *w* for which Count(the, *w*) = 0

| $x$ | $\text{Count}(x)$ | $\text{Count}^*(x)$ | $\frac{\text{Count}^*(x)}{\text{Count}(x)}$ |
|---|---|---|---|
| the | 48 | | |
| the, dog | 15 | 14.5 | 14.5/48 |
| the, woman | 11 | 10.5 | 10.5/48 |
| the, man | 10 | 9.5 | 9.5/48 |
| the, park | 5 | 4.5 | 4.5/48 |
| the, job | 2 | 1.5 | 1.5/48 |
| the, telescope | 1 | 0.5 | 0.5/48 |
| the, manual | 1 | 0.5 | 0.5/48 |
| the, afternoon | 1 | 0.5 | 0.5/48 |
| the, country | 1 | 0.5 | 0.5/48 |
| the, street | 1 | 0.5 | 0.5/48 |

# Absolute Discounting

- Define Count*(x) = Count(x) - 0.5

- Missing probability mass:

$$\alpha(w_{i-1}) = 1 - \sum_{w} \frac{\text{Count}^*(w_{i-1,w})}{\text{Count}(w_{i-1})}$$

$$\alpha(\text{the}) = 10 \times 0.5/48 = 5/48$$

- Divide this mass between words *w* for which Count(the, *w*) = 0

| $x$ | Count$(x)$ | Count$^*(x)$ | $\frac{\text{Count}^*(x)}{\text{Count}(x)}$ |
|---|---|---|---|
| the | 48 | | |
| the, dog | 15 | 14.5 | 14.5/48 |
| the, woman | 11 | 10.5 | 10.5/48 |
| the, man | 10 | 9.5 | 9.5/48 |
| the, park | 5 | 4.5 | 4.5/48 |
| the, job | 2 | 1.5 | 1.5/48 |
| the, telescope | 1 | 0.5 | 0.5/48 |
| the, manual | 1 | 0.5 | 0.5/48 |
| the, afternoon | 1 | 0.5 | 0.5/48 |
| the, country | 1 | 0.5 | 0.5/48 |
| the, street | 1 | 0.5 | 0.5/48 |

# Absolute Discounting

| $x$ | Count($x$) | Count$^*$($x$) | $\frac{\text{Count}^*(x)}{\text{Count}(x)}$ |
|---|---|---|---|
| the | 48 | | |
| the, dog | 15 | 14.5 | 14.5/48 |
| the, woman | 11 | 10.5 | 10.5/48 |
| the, man | 10 | 9.5 | 9.5/48 |
| the, park | 5 | 4.5 | 4.5/48 |
| the, job | 2 | 1.5 | 1.5/48 |
| the, telescope | 1 | 0.5 | 0.5/48 |
| the, manual | 1 | 0.5 | 0.5/48 |
| the, afternoon | 1 | 0.5 | 0.5/48 |
| the, country | 1 | 0.5 | 0.5/48 |
| the, street | 1 | 0.5 | 0.5/48 |

$$\alpha(\text{the}) = 10 \times 0.5/48 = 5/48$$

$$P_{\text{abs\_discount}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} \quad \text{if } c(w_{i-1}, w_i) > 0$$

for all $w'$ s.t. $c(w_{i-1}, w') = 0$ if $c(w_{i-1}, w_i) = 0$ $\alpha(w_{i-1}) \frac{P(w_i)}{\sum_{w'} P(w')}$

Unigram probabilities