

N-gram Language Models

COMP90042

Natural Language Processing

Lecture 3

Semester 1 2021 Week 2

Jey Han Lau



THE UNIVERSITY OF
MELBOURNE

Language Models

- One application NLP is about *explaining language*
 - ▶ Why some sentences are more **fluent** than others
- E.g. in speech recognition:
 - *recognise speech* > *wreck a nice beach*
- We measure ‘goodness’ using **probabilities** estimated by language models
- Language model can also be used for **generation**

Talk to Transformer

See how a modern neural network completes your text. Type a custom snippet or try one of the examples. This is a limited demo of [InferKit](#).

Custom prompt ▼

Donald Trump decides to go vegan.

Generate Another

Completion

Donald Trump decides to go vegan.

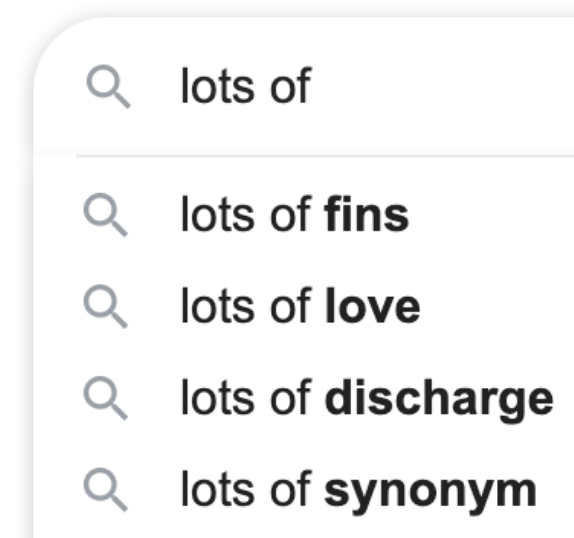
For a man who supposedly prefers steak and KFC over salad and who once tweeted "Coffee and Donald Trump are not mutually exclusive" we have to admit it's an unusual move.

But that's because of what it really symbolises. For while his choice to swap his meat-heavy diet for a vegan one is probably not a death knell for his career, it is a clear sign that his brand of politics, Donald Trump, is running out of steam.

It wasn't too long ago that the President of the United States could always be

Language Models

- Useful for
 - ▶ Query completion
 - ▶ Optical character recog.
- Other generation tasks
 - ▶ Machine translation
 - ▶ Summarisation
 - ▶ Dialogue systems
- Nowadays pretrained language models are the backbone of modern NLP systems



Outline

- Deriving n -gram language models
- Smoothing to deal with sparsity
- Generating language

Probabilities: Joint to Conditional

Our goal is to get a probability for an arbitrary sequence of m words

$$P(w_1, w_2, \dots, w_m)$$

First step is to apply the chain rule to convert joint probabilities to conditional ones

$$P(w_1, w_2, \dots, w_m) = P(w_1)P(w_2 | w_1)P(w_3 | w_1, w_2) \dots \\ P(w_m | w_1, \dots, w_{m-1})$$

The Markov Assumption

Still intractable, so make a simplifying assumption:

$$P(w_i | w_1 \dots w_{i-1}) \approx P(w_i | w_{i-n+1} \dots w_{i-1})$$

For some small n

When $n = 1$, a unigram model

$$P(w_1, w_2, \dots, w_m) = \prod_{i=1}^m P(w_i)$$

the dog ^{w_i} barks

When $n = 2$, a bigram model

$$P(w_1, w_2, \dots, w_m) = \prod_{i=1}^m P(w_i | w_{i-1})$$

the ^{w_i} dog barks

When $n = 3$, a trigram model

$$P(w_1, w_2, \dots, w_m) = \prod_{i=1}^m P(w_i | w_{i-2} w_{i-1})$$

the ^{w_i} dog barks

Maximum Likelihood Estimation

How do we calculate the probabilities? Estimate based on counts in our corpus:

For unigram models,

$$P(w_i) = \frac{C(w_i)}{M} \qquad \frac{C(\text{barks})}{M}$$

For bigram models,

$$P(w_i | w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})} \qquad \frac{C(\text{dog barks})}{C(\text{dog})}$$

For n-gram models generally,

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

Book-ending Sequences

- Special tags used to denote start and end of sequence
 - ▶ `<s>` = sentence start
 - ▶ `</s>` = sentence end

Trigram example

Corpus:

yes no no no no yes
no no no yes yes yes no

What is the probability of

yes no no yes

under a trigram language model?

$P(\text{yes no no yes}) =$

$P(\text{yes} \mid \langle s \rangle \langle s \rangle) \times$

$P(\text{no} \mid \langle s \rangle \text{yes}) \times$

$P(\text{no} \mid \text{yes no}) \times$

$P(\text{yes} \mid \text{no no}) \times$

$P(\langle /s \rangle \mid \text{no yes})$

 Need to predict $\langle /s \rangle$ because it's the end of sentence!

Corpus:

<s> <s> yes no no no no yes </s>

<s> <s> no no no yes yes yes no </s>

Compute: $P(\text{yes no no yes})$

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

Corpus:

<s> <s> yes no no no no yes </s>

<s> <s> no no no yes yes yes no </s>

Compute: $P(\text{yes no no yes})$

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

$P(\text{yes} | \text{<s> <s>})$

1/2

<s> <s> yes

<s> <s> no

Corpus:

<s> **<s> yes no** no no no yes </s>

<s> <s> no no no yes yes yes no </s>

Compute: $P(\text{yes no no yes})$

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

$P(\text{yes} \text{<s> <s>})$	1/2	<s> <s> yes <s> <s> no
$P(\text{no} \text{<s> yes})$	1/1	<s> yes no

Corpus:

<s> <s> *yes no no* no no yes </s>
 <s> <s> no no no yes yes *yes no* </s>

Compute: $P(\text{yes no no yes})$

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

$P(\text{yes} \text{<s> <s>})$	1/2	<s> <s> <i>yes</i> <s> <s> <i>no</i>
$P(\text{no} \text{<s> yes})$	1/1	<s> <i>yes no</i>
$P(\text{no} \text{yes no})$	1/2	<i>yes no no</i> <i>yes no </s></i>

Corpus:

<s> <s> yes no no no no yes </s>

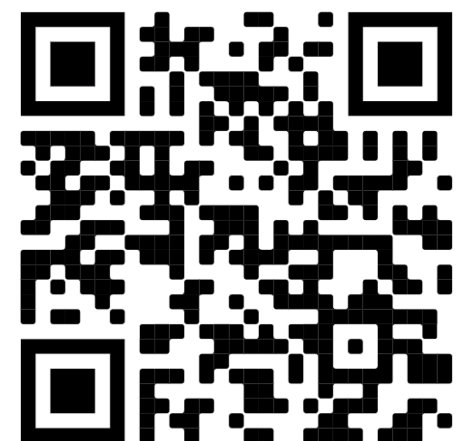
<s> <s> no no no yes yes yes no </s>

Compute: $P(\text{yes no no yes})$

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

$P(\text{yes} \text{<s> <s>})$	1/2	<i><s> <s> yes <s> <s> no</i>
$P(\text{no} \text{<s> yes})$	1/1	<i><s> yes no</i>
$P(\text{no} \text{yes no})$	1/2	<i>yes no no yes no </s></i>
$P(\text{yes} \text{no no})$?	

[PollEv.com/jeyhanlau569](https://poll-ev.com/jeyhanlau569)



$P(\text{yes} \mid \text{no no}) = ?$



Corpus:

<s> <s> yes no no no no yes </s>

<s> <s> no no no yes yes yes no </s>

Compute: $P(\text{yes no no yes})$

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

$P(\text{yes} \text{<s> <s>})$	1/2	<i><s> <s> yes <s> <s> no</i>
$P(\text{no} \text{<s> yes})$	1/1	<i><s> yes no</i>
$P(\text{no} \text{yes no})$	1/2	<i>yes no no yes no </s></i>
$P(\text{yes} \text{no no})$	2/5	<i>no no no no no no no no yes no no no no no yes</i>

Corpus:

<s> <s> yes no no no no yes </s>

<s> <s> no no no yes yes yes no </s>

Compute: $P(\text{yes no no yes})$

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

$P(\text{yes} \mid \text{<s> <s>})$	1/2	<i><s> <s> yes <s> <s> no</i>
$P(\text{no} \mid \text{<s> yes})$	1/1	<i><s> yes no</i>
$P(\text{no} \mid \text{yes no})$	1/2	<i>yes no no yes no </s></i>
$P(\text{yes} \mid \text{no no})$	2/5	<i>no no no no no no no no yes no no no no no yes</i>
$P(\text{</s>} \mid \text{no yes})$	1/2	<i>no yes </s> no yes yes</i>

Corpus:

<s> <s> yes no no no no yes </s>

<s> <s> no no no yes yes yes no </s>

Compute: $P(\text{yes no no yes}) = \frac{1}{2} \times 1 \times \frac{1}{2} \times \frac{2}{5} \times \frac{1}{2} = 0.05$

$P(\text{yes} \mid \text{<s> <s>})$	$\frac{1}{2}$	<i><s> <s> yes <s> <s> no</i>
$P(\text{no} \mid \text{<s> yes})$	$\frac{1}{1}$	<i><s> yes no</i>
$P(\text{no} \mid \text{yes no})$	$\frac{1}{2}$	<i>yes no no yes no </s></i>
$P(\text{yes} \mid \text{no no})$	$\frac{2}{5}$	<i>no no no no no no no no yes no no no no no yes</i>
$P(\text{</s>} \mid \text{no yes})$	$\frac{1}{2}$	<i>no yes </s> yes no </s></i>

Several Problems

- Language has long distance effects — need large n
 - ▶ The *lecture/s* that took place last week *was/were* on preprocessing.
- Resulting probabilities are often very small
 - ▶ Use log probability to avoid numerical underflow
- What about unseen words?
 - ▶ Special symbol to represent them (e.g. <UNK>)
- Unseen n-grams?
 - ▶ $P(w_1, w_2, \dots, w_m) = P(w_1 \mid \langle s \rangle) \times P(w_2 \mid w_1) \dots$
 - ▶ Need to smooth the LM!

whole term = 0 if $P(w_2|w_1) = 0$



Smoothing

Smoothing

- Basic idea: give events you've never seen before some probability
- Must be the case that $P(\text{everything}) = 1$
- Many different kinds of smoothing
 - ▶ Laplacian (add-one) smoothing
 - ▶ Add- k smoothing
 - ▶ Absolute discounting
 - ▶ Kneser-Ney
 - ▶ And others...

Laplacian (Add-one) Smoothing

- Simple idea: pretend we've seen each n -gram once more than we did.

For unigram models (\mathbf{V} = the vocabulary),

$$P_{add1}(w_i) = \frac{C(w_i) + 1}{M + |\mathbf{V}|}$$

For bigram models,

$$P_{add1}(w_i | w_{i-1}) = \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |\mathbf{V}|}$$

Add-one Example

<s> the rat ate the cheese </s>

What's the bigram probability $P(\text{ate} \mid \text{rat})$ under add-one smoothing?

$$= \frac{C(\text{rat ate}) + 1}{C(\text{rat}) + |V|} = \frac{2}{6}$$

$V = \{ \text{the, rat, ate, cheese, } \langle /s \rangle \}$

What's the bigram probability $P(\text{ate} \mid \text{cheese})$ under add-one smoothing?

$$= \frac{C(\text{cheese ate}) + 1}{C(\text{cheese}) + |V|} = \frac{1}{6}$$

<s> is not part of vocabulary because we never need to infer its conditional probability (e.g. $P(\langle s \rangle \mid \dots)$)

Recall: $P(\text{yes no no yes}) =$
 $P(\text{yes} \mid \langle s \rangle \langle s \rangle) \times P(\text{no} \mid \langle s \rangle \text{yes}) \times$
 $P(\text{no} \mid \text{yes no}) \times P(\text{yes} \mid \text{no no}) \times$
 $P(\langle /s \rangle \mid \text{no yes})$

Add- k Smoothing

- Adding one is often too much
- Instead, add a fraction k
- AKA Lidstone Smoothing

$$P_{addk}(w_i | w_{i-1}, w_{i-2}) = \frac{C(w_{i-2}, w_{i-1}, w_i) + k}{C(w_{i-2}, w_{i-1}) + k |V|}$$

- Have to choose k

Lidstone Smoothing

Context = *alleged*

- 5 observed bi-grams
- 2 unobserved bi-grams

			Lidstone smoothing, $\alpha = 0.1$	
	counts	unsmoothed probability	effective counts	smoothed probability
<i>impropriety</i>	8	0.4	7.826	0.391
<i>offense</i>	5	0.25	4.928	0.246
<i>damage</i>	4	0.2	3.961	0.198
<i>deficiencies</i>	2	0.1	2.029	0.101
<i>outbreak</i>	1	0.05	1.063	0.053
<i>infirmity</i>	0	0	0.097	0.005
<i>alleged</i>	0	0	0.097	0.005
	20	1.0	20	1.0

$$0.391 \times 20$$

$$(0 + 0.1) / (20 + 7 \times 0.1)$$

$$(8 + 0.1) / (20 + 7 \times 0.1)$$

Absolute Discounting

- ‘Borrows’ a **fixed** probability mass from observed n-gram counts
- Redistributes it to unseen n-grams

Absolute Discounting

Context = *alleged*

- 5 observed bi-grams
- 2 unobserved bi-grams

			Lidstone smoothing, $\alpha = 0.1$		Discounting, $d = 0.1$	
	counts	unsmoothed probability	effective counts	smoothed probability	effective counts	smoothed probability
<i>impropriety</i>	8	0.4	7.826	0.391	7.9	0.395
<i>offense</i>	5	0.25	4.928	0.246	4.9	0.245
<i>damage</i>	4	0.2	3.961	0.198	3.9	0.195
<i>deficiencies</i>	2	0.1	2.029	0.101	1.9	0.095
<i>outbreak</i>	1	0.05	1.063	0.053	0.9	0.045
<i>infirmity</i>	0	0	0.097	0.005	0.25	0.013
<i>alleged</i>	0	0	0.097	0.005	0.25	0.013
	20	1.0	20	1.0	20	1.0

$8 - 0.1$ $(0.1 \times 5) / 2$ $0.25 / 20$

Backoff

- Absolute discounting redistributes the probability mass **equally** for all unseen n-grams
- Katz Backoff: redistributes the mass based on a **lower order model** (e.g. unigram)

$$P_{katz}(w_i|w_{i-1}) = \begin{cases} \frac{C(w_{i-1}, w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}, w_i) > 0 \\ \alpha(w_{i-1}) \times \frac{P(w_i)}{\sum_{w_j: C(w_{i-1}, w_j)=0} P(w_j)}, & \text{otherwise} \end{cases}$$

sum unigram probabilities for all words that do not co-occur with context w_{i-1}
 e.g. $P(\text{infirmity}) + P(\text{alleged})$

unigram probability for w_i
 e.g. $P(\text{infirmity})$

the amount of probability mass that has been discounted for context w_{i-1}
 ($(0.1 \times 5) / 20$ in previous slide)

the amount of probability mass that has been discounted for context w_{i-1}
 ($(0.1 \times 5) / 20$ in previous slide)

Issues with Katz Backoff

$$P_{katz}(w_i|w_{i-1}) = \begin{cases} \frac{C(w_{i-1}, w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}, w_i) > 0 \\ \alpha(w_{i-1}) \times \frac{P(w_i)}{\sum_{w_j: C(w_{i-1}, w_j)=0} P(w_j)}, & \text{otherwise} \end{cases}$$

- *I can't see without my reading ____*
- $C(\text{reading}, \text{glasses}) = C(\text{reading}, \text{Francisco}) = 0$
- $C(\text{Francisco}) > C(\text{glasses})$
- Katz backoff will give higher probability to *Francisco*

Kneser-Ney Smoothing

- Redistribute probability mass based on the **versatility** of the lower order n-gram
- AKA “continuation probability”
- What is versatility?
 - ▶ High versatility -> co-occurs with a lot of unique words, e.g. glasses
 - men’s glasses, black glasses, buy glasses, etc
 - ▶ Low versatility -> co-occurs with few unique words, e.g. francisco
 - san francisco

Kneser-Ney Smoothing

$$P_{KN}(w_i | w_{i-1}) = \begin{cases} \frac{C(w_{i-1}, w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}, w_i) > 0 \\ \beta(w_{i-1})P_{cont}(w_i), & \text{otherwise} \end{cases}$$

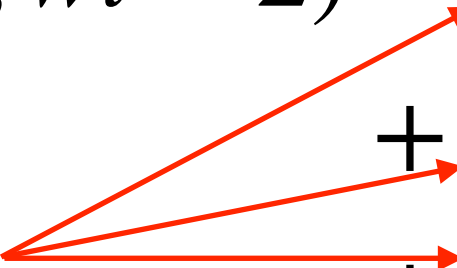
the amount of probability mass that
has been discounted for context w_{i-1}

$$P_{cont}(w_i) = \frac{|\{w_{i-1} : C(w_{i-1}, w_i) > 0\}|}{\sum_{w_i} |\{w_{i-1} : C(w_{i-1}, w_i) > 0\}|}$$

- Intuitively the numerator of P_{cont} counts the number of unique w_{i-1} that co-occurs with w_i
- High continuation counts for glasses
- Low continuation counts for Franciso

Interpolation

- A better way to combine different orders of n -gram models
- Weighted sum of probabilities across progressively shorter contexts
- Interpolated trigram model:

$$P_{IN}(w_i | w_{i-1}, w_{i-2}) = \lambda_3 P_3(w_i | w_{i-2}, w_{i-1})$$
$$+ \lambda_2 P_2(w_i | w_{i-1})$$
$$+ \lambda_1 P_1(w_i)$$


Learned based on held out data

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

Interpolated Kneser-Ney Smoothing

- Interpolation instead of back-off

$$P_{IKN}(w_i | w_{i-1}) = \frac{C(w_{i-1}, w_i) - D}{C(w_{i-1})} + \gamma(w_{i-1})P_{cont}(w_i)$$

- where $\gamma(w_{i-1})$ is normalising constant such that
$$\sum_{w_{i-1} \in V} P_{IKN}(w_i | w_{i-1}) = 1.0$$

In Practice

- Commonly used Kneser-Ney language models use 5-grams as max order
- Has different discount values for each n-gram order.

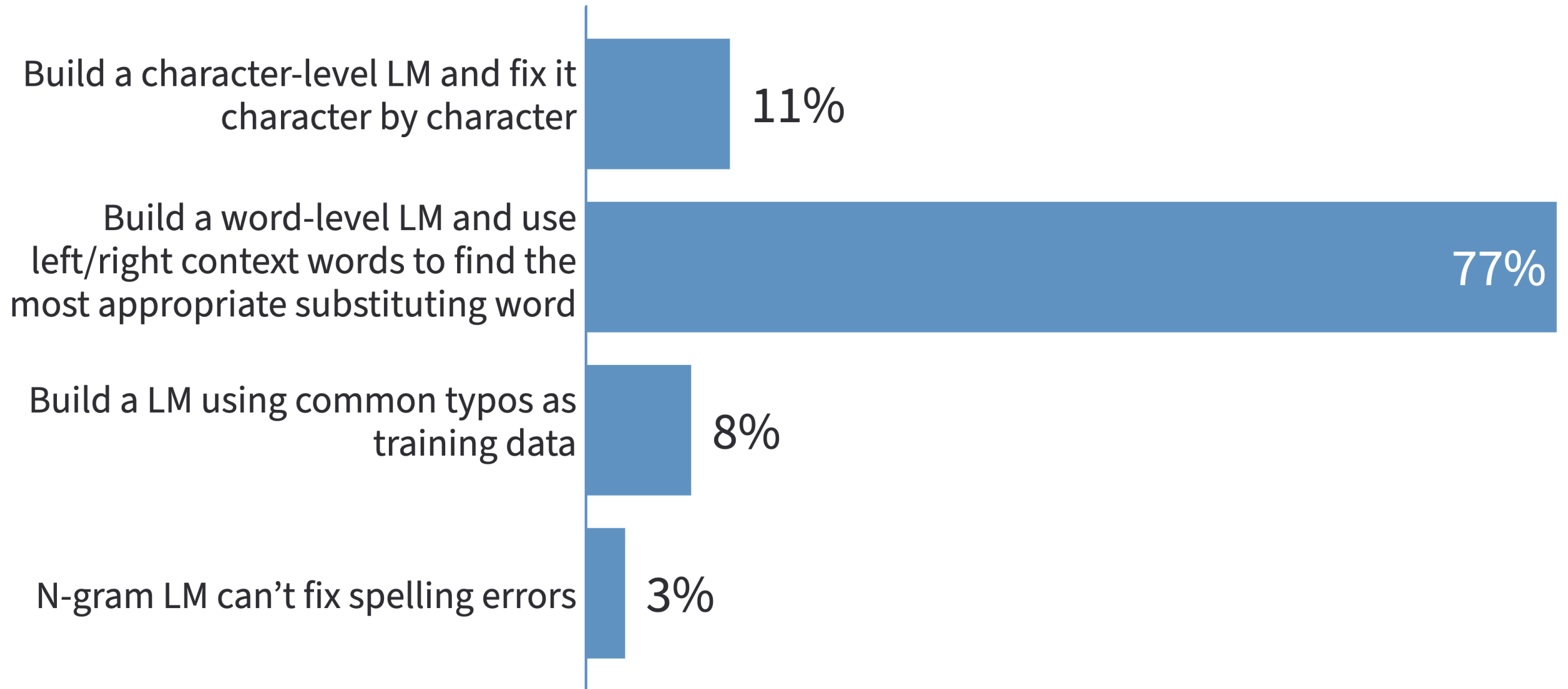
How can we use n -gram language model to fix spelling errors?

- Build a character-level LM and fix it character by character
- Build a word-level LM and use left/right context words to find the most appropriate substituting word
- Build a LM using common typos as training data
- N-gram LM can't fix spelling errors

PollEv.com/jeyhanlau569



How can we use n-gram language model to fix spelling errors?



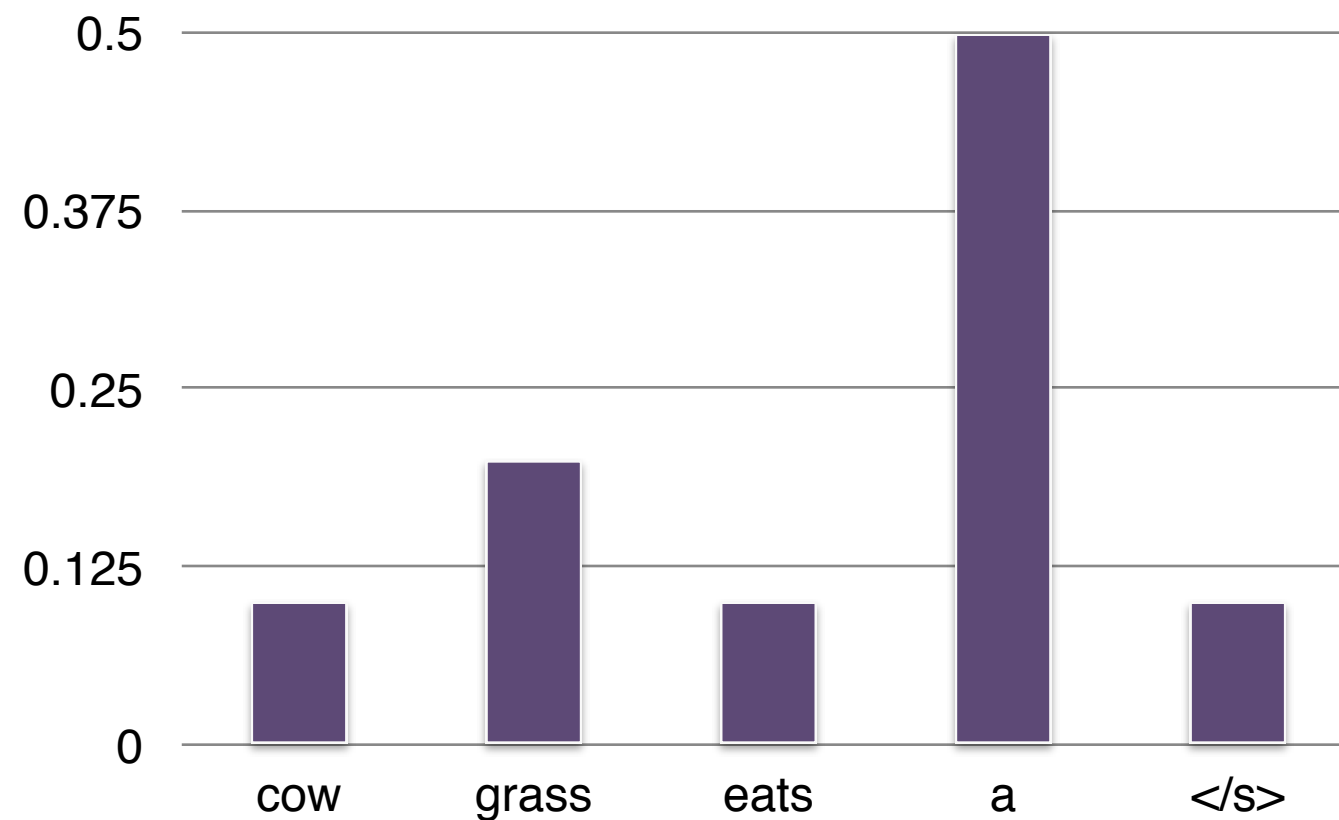
Generating Language

Generation

- Given an initial word, draw the next word according to the probability distribution produced by the language model.
- Include $(n-1)$ `<s>` tokens for n -gram model, to provide context to generate first word
 - ▶ never generate `<s>`
 - ▶ generating `</s>` terminates the sequence

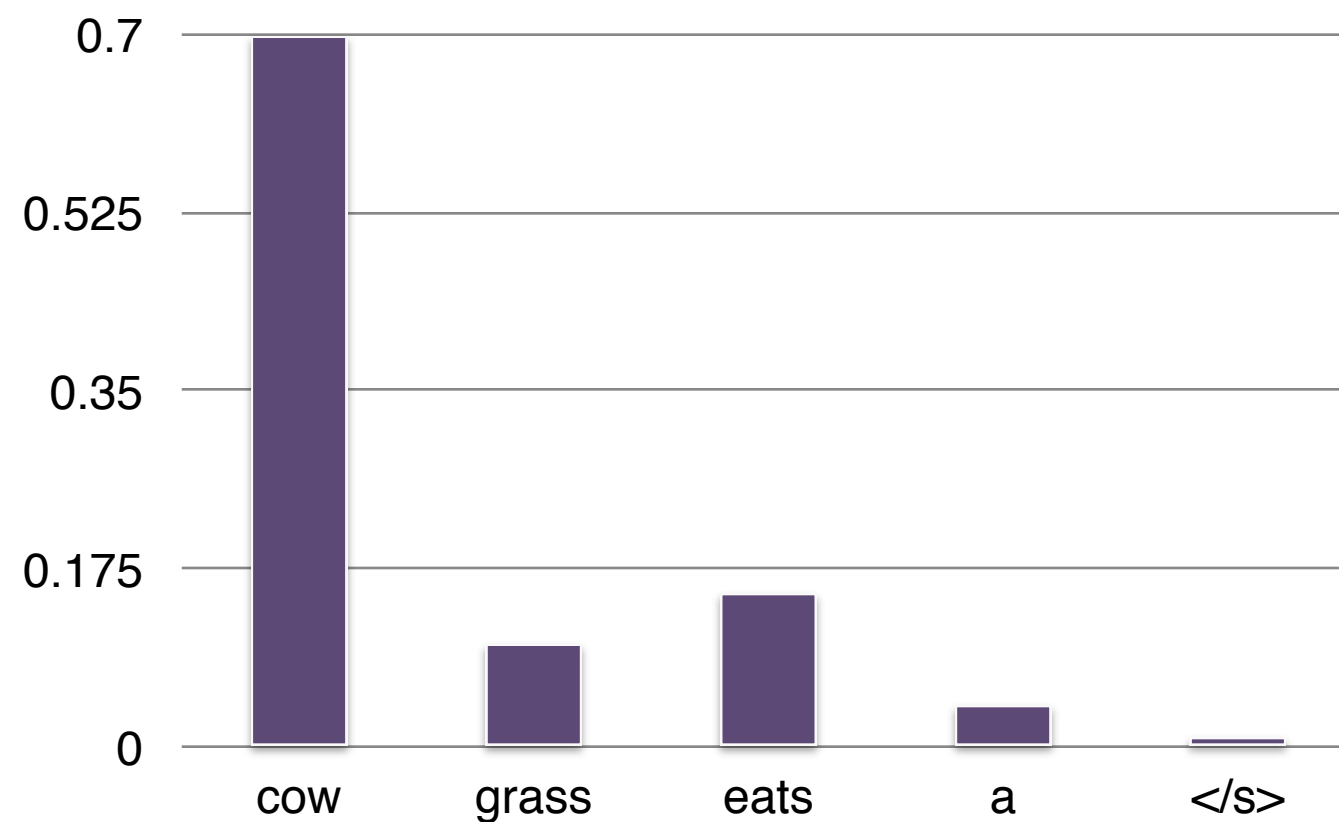
Generation (Bigram LM)

- Sentence = $\langle s \rangle$
- $P(? \mid \langle s \rangle) = \text{"a"}$



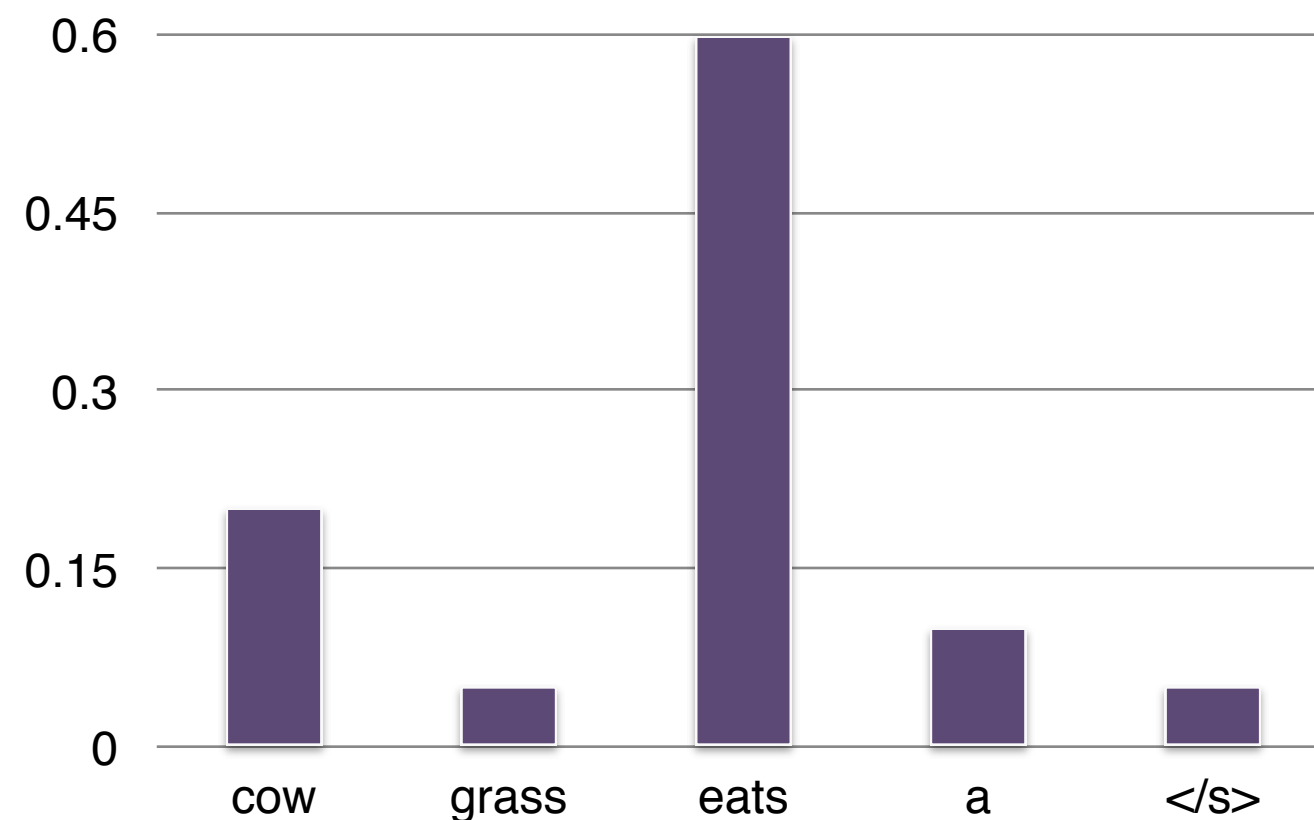
Generation (Bigram LM)

- Sentence = $\langle s \rangle a$
- $P(? \mid a) = \text{"cow"}$



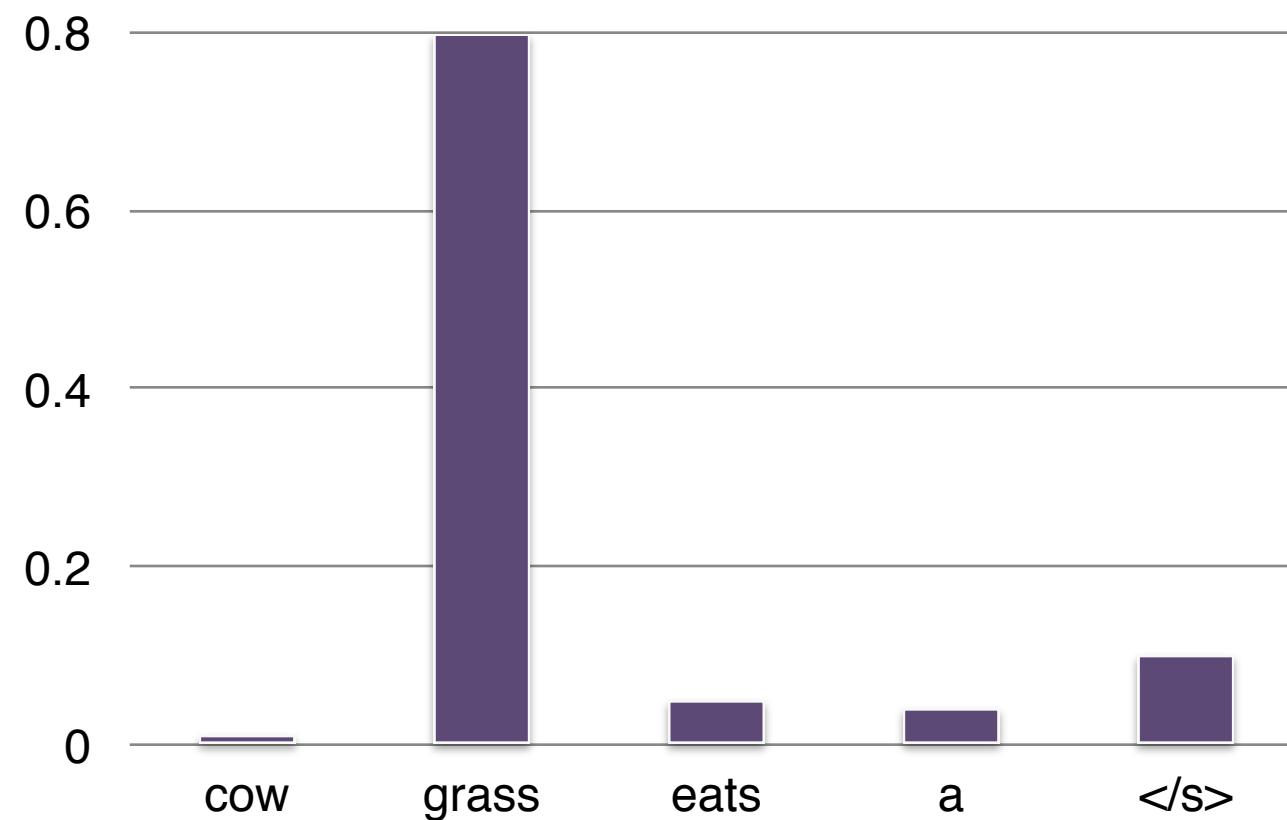
Generation (Bigram LM)

- Sentence = $\langle s \rangle$ *a cow*
- $P(? \mid \text{cow}) = \text{"eats"}$



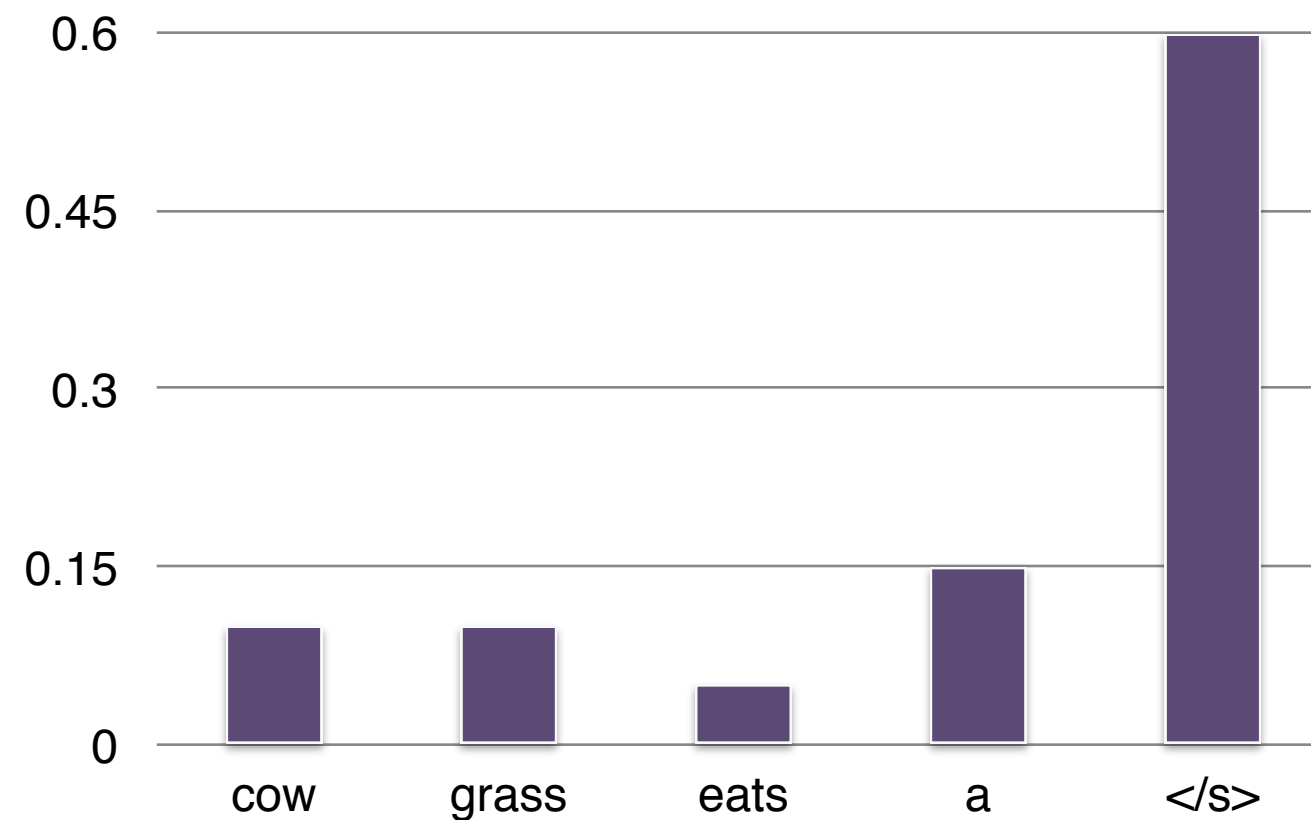
Generation (Bigram LM)

- Sentence = $\langle s \rangle$ *a cow eats*
- $P(? \mid \text{eats}) = \text{"grass"}$



Generation (Bigram LM)

- Sentence = $\langle s \rangle$ *a cow eats grass*
- $P(? \mid \text{grass}) = \langle /s \rangle$



Generation (Bigram LM)

- Sentence = $\langle s \rangle$ *a cow eats grass* $\langle /s \rangle$
- Done!

How to Select Next Word?

- Argmax: takes highest probability word each turn
 - ▶ Greedy search
- Beam search decoding (lecture 17):
 - ▶ Keeps track of top-N highest probability words each turn
 - ▶ Select sequence of words that produce the best sentence probability
- Randomly samples from the distribution

A Final Word

- *N*-gram language models are a simple but effective way to capture the predictability of language
- Can be trained in an unsupervised fashion, scalable to large corpora
- Require smoothing to be effective
- Modern language models uses neural networks (lecture 8)

Reading

- E18 Chapter 6 (skip 6.3)