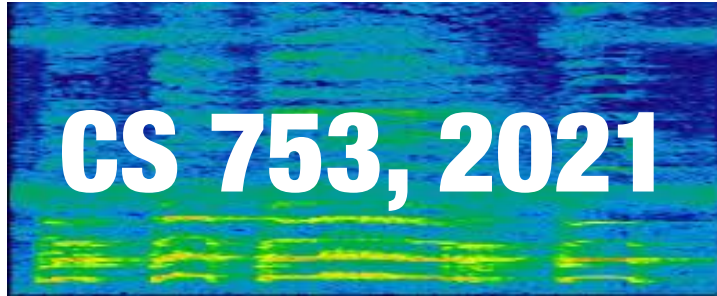


Language Models (I)

Lecture 4c



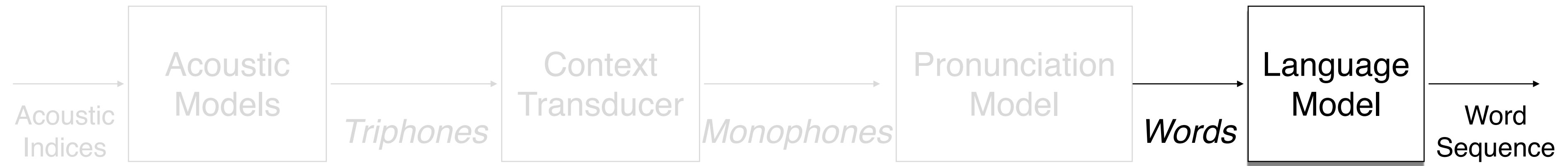
CS 753, 2021

Instructor: Preethi Jyothi, IITB

Next, language models



Next, language models



- Language models
- provide information about word reordering

$\text{Pr}(\text{"she class taught a"}) < \text{Pr}(\text{"she taught a class"})$

Next, language models



- Language models
- provide information about word reordering

$\text{Pr}(\text{"she class taught a"}) < \text{Pr}(\text{"she taught a class"})$

- provide information about the most likely next word

$\text{Pr}(\text{"she taught a class"}) > \text{Pr}(\text{"she taught a speech"})$

Application of language models

- Speech recognition
 - $\text{Pr}(\text{"she taught a class"}) > \text{Pr}(\text{"sheet or tuck lass"})$

Application of language models

- Speech recognition
 - $\text{Pr}(\text{"she taught a class"}) > \text{Pr}(\text{"sheet or tuck lass"})$
- Machine translation

Application of language models

- Speech recognition
 - $\text{Pr}(\text{"she taught a class"}) > \text{Pr}(\text{"sheet or tuck lass"})$
- Machine translation
- Handwriting recognition/Optical character recognition

Application of language models

- Speech recognition
 - $\text{Pr}(\text{"she taught a class"}) > \text{Pr}(\text{"sheet or tuck lass"})$
- Machine translation
- Handwriting recognition/Optical character recognition
- Spelling correction of sentences

Application of language models

- Speech recognition
 - $\text{Pr}(\text{"she taught a class"}) > \text{Pr}(\text{"sheet or tuck lass"})$
- Machine translation
- Handwriting recognition/Optical character recognition
- Spelling correction of sentences
- Summarization, dialog generation, information retrieval, etc.

Popular Language Modelling Toolkits

- SRILM Toolkit:

<http://www.speech.sri.com/projects/srilm/>

- KenLM Toolkit:

<https://kheafield.com/code/kenlm/>

- OpenGrm NGram Library:

<http://opengrm.org/>

Introduction to probabilistic LMs

Probabilistic or Statistical Language Models

- Given a word sequence, $W = \{w_1, \dots, w_n\}$, what is $\Pr(W)$?
- Decompose $\Pr(W)$ using the chain rule:

$$\Pr(w_1, w_2, \dots, w_{n-1}, w_n) = \underbrace{\Pr(w_1)} \underbrace{\Pr(w_2|w_1)} \underbrace{\Pr(w_3|w_1, w_2)} \dots \underbrace{\Pr(w_n|w_1, \dots, w_{n-1})}$$

- Sparse data with long word contexts: How do we estimate the probabilities $\Pr(w_n|w_1, \dots, w_{n-1})$?

Estimating word probabilities

Estimating word probabilities

- Accumulate counts of words and word contexts
- Compute normalised counts to get next-word probabilities

Estimating word probabilities

- Accumulate counts of words and word contexts
- Compute normalised counts to get next-word probabilities

- E.g. $\Pr(\text{"class" | she taught a"})$
$$= \frac{\pi(\text{"she taught a class"})}{\pi(\text{"she taught a"})}$$

where $\pi(\text{"..."})$ refers to counts derived from a large English text corpus

Estimating word probabilities

- Accumulate counts of words and word contexts
- Compute normalised counts to get next-word probabilities
- E.g. $\Pr(\text{"class" | she taught a"})$
$$= \frac{\pi(\text{"she taught a class"})}{\pi(\text{"she taught a"})}$$

where $\pi(\text{"..."})$ refers to counts derived from a large English text corpus

- What is the obvious limitation here?

Estimating word probabilities

- Accumulate counts of words and word contexts
- Compute normalised counts to get next-word probabilities

- E.g. $\Pr(\text{"class" | she taught a"})$
$$= \frac{\pi(\text{"she taught a class"})}{\pi(\text{"she taught a"})}$$

where $\pi(\text{"..."})$ refers to counts derived from a large English text corpus

- What is the obvious limitation here? **We'll never see enough data**

Simplifying Markov Assumption

Simplifying Markov Assumption

- Markov chain:
 - Limited memory of previous word history: Only last m words are included

Simplifying Markov Assumption

- Markov chain:
 - Limited memory of previous word history: Only last m words are included
- 1-order language model (or bigram model)

$$\Pr(w_1, w_2, \dots, w_{n-1}, w_n) \cong \Pr(w_1 | \langle s \rangle) \underbrace{\Pr(w_2 | w_1)} \underbrace{\Pr(w_3 | w_2)} \dots \underbrace{\Pr(w_n | w_{n-1})}$$

Simplifying Markov Assumption

- Markov chain:
 - Limited memory of previous word history: Only last m words are included
- 1-order language model (or bigram model)

$$\Pr(w_1, w_2, \dots, w_{n-1}, w_n) \cong \Pr(w_1 | \langle s \rangle) \Pr(w_2 | w_1) \Pr(w_3 | w_2) \dots \Pr(w_n | w_{n-1})$$

Simplifying Markov Assumption

- Markov chain:
 - Limited memory of previous word history: Only last m words are included

- 1-order language model (or bigram model)

$$\Pr(w_1, w_2, \dots, w_{n-1}, w_n) \cong \Pr(w_1 | \langle s \rangle) \Pr(w_2 | w_1) \Pr(w_3 | w_2) \dots \Pr(w_n | w_{n-1})$$

- 2-order language model (or trigram model)

$$\Pr(w_1, w_2, \dots, w_{n-1}, w_n) \cong \Pr(w_2 | w_1, \langle s \rangle) \Pr(w_3 | w_1, w_2) \dots \Pr(w_n | w_{n-2}, w_{n-1})$$

Simplifying Markov Assumption

- Markov chain:
 - Limited memory of previous word history: Only last m words are included

- 1-order language model (or bigram model)

$$\Pr(w_1, w_2, \dots, w_{n-1}, w_n) \cong \Pr(w_1 | \langle s \rangle) \Pr(w_2 | w_1) \Pr(w_3 | w_2) \dots \Pr(w_n | w_{n-1})$$

- 2-order language model (or trigram model)

$$\Pr(w_1, w_2, \dots, w_{n-1}, w_n) \cong \Pr(w_2 | w_1, \langle s \rangle) \Pr(w_3 | w_1, w_2) \dots \Pr(w_n | w_{n-2}, w_{n-1})$$

- Ngram model is an N-1th order Markov model

Estimating Ngram Probabilities

Estimating Ngram Probabilities

- Maximum Likelihood Estimates
- Unigram model

$$\Pr_{ML}(w_1) = \frac{\pi(w_1)}{\sum_i \pi(w_i)}$$

Estimating Ngram Probabilities

- Maximum Likelihood Estimates
- Unigram model

$$\Pr_{ML}(w_1) = \frac{\pi(w_1)}{\sum_i \pi(w_i)}$$

- Bigram model

Estimating Ngram Probabilities

- Maximum Likelihood Estimates
- Unigram model

$$\text{Pr}_{ML}(w_1) = \frac{\pi(w_1)}{\sum_i \pi(w_i)}$$

- Bigram model

$$\text{Pr}_{ML}(w_2|w_1) = \frac{\pi(w_1, w_2)}{\sum_i \pi(w_1, w_i)} \quad \underline{\neq} \pi(w_1)$$

Example

The dog chased a cat
The cat chased away a mouse
The mouse eats cheese



What is $\Pr(\text{"The cat chased a mouse"})$ using a bigram model?

Example

The dog chased a cat
The cat chased away a mouse
The mouse eats cheese

What is $\Pr(\text{"The cat chased a mouse"})$ using a bigram model?

$$\Pr(\text{"<s> The cat chased a mouse </s>"}) =$$

$$\Pr(\text{"The|<s>"}) \cdot \Pr(\text{"cat|The"}) \cdot \Pr(\text{"chased|cat"}) \cdot \Pr(\text{"a|chased"}) \cdot \Pr(\text{"mouse|a"}) \cdot \Pr(\text{"</s>|mouse"}) =$$

$$\frac{3}{3} \cdot \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \underline{\underline{\frac{1}{48}}}$$

Example

The dog chased a cat
The cat chased away a mouse
The mouse eats cheese

What is $\Pr(\text{"The dog eats cheese"})$ using a bigram model?

Example

The dog chased a cat
The cat chased away a mouse
The mouse eats cheese

What is $\Pr(\text{"The dog eats cheese"})$ using a bigram model?

$\Pr(\text{"<s> The dog eats cheese </s>"}) =$

$\Pr(\text{"The|<s>"}) \cdot \Pr(\text{"dog|The"}) \cdot \Pr(\text{"eats|dog"}) \cdot \Pr(\text{"cheese|eats"}) \cdot \Pr(\text{"</s>|cheese"}) =$

$\frac{3}{3} \cdot \frac{1}{3} \cdot \frac{0}{1} \cdot \frac{1}{1} \cdot \frac{1}{1} = 0!$

Example

The dog chased a cat
The cat chased away a mouse
The mouse eats cheese

What is $\Pr(\text{"The dog eats cheese"})$ using a bigram model?

$\Pr(\text{"<s> The dog eats cheese </s>"}) =$

$\Pr(\text{"The|<s>"}) \cdot \Pr(\text{"dog|The"}) \cdot \Pr(\text{"eats|dog"}) \cdot \Pr(\text{"cheese|eats"}) \cdot \Pr(\text{"</s>|cheese"}) =$

$\frac{3}{3} \cdot \frac{1}{3} \cdot \frac{0}{1} \cdot \frac{1}{1} \cdot \frac{1}{1} = 0!$ Due to unseen bigrams

Example

The dog chased a cat
The cat chased away a mouse
The mouse eats cheese

What is $\Pr(\text{"The dog eats cheese"})$ using a bigram model?

$\Pr(\text{"<s> The dog eats cheese </s>"}) =$

$\Pr(\text{"The|<s>"}) \cdot \Pr(\text{"dog|The"}) \cdot \Pr(\text{"eats|dog"}) \cdot \Pr(\text{"cheese|eats"}) \cdot \Pr(\text{"</s>|cheese"}) =$

$\frac{3}{3} \cdot \frac{1}{3} \cdot \frac{0}{1} \cdot \frac{1}{1} \cdot \frac{1}{1} = 0!$ Due to unseen bigrams

How do we deal with unseen bigrams? We'll come back to it.

Open vs. closed vocabulary task

Open vs. closed vocabulary task

- Closed vocabulary task: Use a fixed vocabulary, V. We know all the words in advance.

Open vs. closed vocabulary task

- Closed vocabulary task: Use a fixed vocabulary, V . We know all the words in advance.
- More realistic setting, we don't know all the words in advance. Open vocabulary task. Encounter out-of-vocabulary (OOV) words during test time.

Open vs. closed vocabulary task

- Closed vocabulary task: Use a fixed vocabulary, V . We know all the words in advance.
- More realistic setting, we don't know all the words in advance. Open vocabulary task. Encounter out-of-vocabulary (OOV) words during test time.
- Create an unknown word: $\langle \text{UNK} \rangle$
 - Estimating $\langle \text{UNK} \rangle$ probabilities: Determine a vocabulary V . Change all words in the training set not in V to $\langle \text{UNK} \rangle$
 - Now train its probabilities like a regular word
 - At test time, use $\langle \text{UNK} \rangle$ probabilities for words not in training

Evaluating Language Models

Evaluating Language Models

- Extrinsic evaluation:
 - To compare Ngram models A and B, use both within a specific speech recognition system (keeping all other components the same)
 - Compare word error rates (WERs) for A and B

Evaluating Language Models

- Extrinsic evaluation:
 - To compare Ngram models A and B, use both within a specific speech recognition system (keeping all other components the same)
 - Compare word error rates (WERs) for A and B
 - Time-consuming process!

Intrinsic Evaluation

Intrinsic Evaluation

- Evaluate the language model in a standalone manner

Intrinsic Evaluation

- Evaluate the language model in a standalone manner
- How likely does the model consider the text in a test set?

Intrinsic Evaluation

- Evaluate the language model in a standalone manner
- How likely does the model consider the text in a test set?
- How closely does the model approximate the actual (test set) distribution?
- Same measure can be used to address both questions — perplexity!

Measures of LM quality

- How likely does the model consider the text in a test set?
- How closely does the model approximate the actual (test set) distribution?
- Same measure can be used to address both questions — perplexity!

Perplexity (I)

Perplexity (I)

- How likely does the model consider the text in a test set?
- $\text{Perplexity}(\text{test}) = 1/\text{Pr}_{\text{model}}[\text{test}]$

Perplexity (I)

- How likely does the model consider the text in a test set?
- $\text{Perplexity}(\text{test}) = 1/\text{Pr}_{\text{model}}[\text{text}]$
- Normalized by text length:
 - $\text{Perplexity}(\text{test}) = (1/\text{Pr}_{\text{model}}[\text{text}])^{1/N}$ where N = number of tokens in test

Perplexity (I)

- How likely does the model consider the text in a test set?
- $\text{Perplexity}(\text{test}) = 1/\text{Pr}_{\text{model}}[\text{text}]$
- Normalized by text length:
 - $\text{Perplexity}(\text{test}) = \underbrace{(1/\text{Pr}_{\text{model}}[\text{text}])^{1/N}}$ where N = number of tokens in test
 - e.g. If model predicts i.i.d. words from a dictionary of size L , per word perplexity = $(1/\underbrace{(1/L)^N})^{1/N} = \underline{\underline{L}}$



Intuition for Perplexity

Intuition for Perplexity

- Shannon's guessing game builds intuition for perplexity
- What is the surprisal factor in predicting the next word?

- At the stall, I had tea and _____

<u>biscuits</u>	0.1
<u>samosa</u>	0.1
coffee	0.01
<u>rice</u>	0.001
⋮	
<u>tree</u>	0.000000001

Intuition for Perplexity

- Shannon's guessing game builds intuition for perplexity
- What is the surprisal factor in predicting the next word?
 - At the stall, I had tea and _____

<i>biscuits</i>	<i>0.1</i>
<i>samosa</i>	<i>0.1</i>
<i>coffee</i>	<i>0.01</i>
<i>rice</i>	<i>0.001</i>
<i>⋮</i>	
<i>tree</i>	<i>0.000000001</i>
- A better language model would assign a higher probability to the actual word that fills the blank (and hence lead to lesser surprisal/perplexity)

Measures of LM quality

- How likely does the model consider the text in a test set?
- How closely does the model approximate the actual (test set) distribution?
- Same measure can be used to address both questions — perplexity!

Perplexity (II)

Perplexity (II)

- How closely does the model approximate the actual (test set) distribution?
- KL-divergence between two distributions X and Y

$$D_{\text{KL}}(X||Y) = \sum_{\sigma} \text{Pr}_X[\sigma] \log (\text{Pr}_X[\sigma]/\text{Pr}_Y[\sigma])$$

Relative entropy
from Y to X

Info. gain if one uses X
instead of Y

Perplexity (II)

- How closely does the model approximate the actual (test set) distribution?
- KL-divergence between two distributions X and Y
$$D_{\text{KL}}(X||Y) = \sum_{\sigma} \text{Pr}_X[\sigma] \log (\text{Pr}_X[\sigma]/\text{Pr}_Y[\sigma])$$
- Equals zero iff $X = Y$; Otherwise, positive

Perplexity (II)

- How closely does the model approximate the actual (test set) distribution?
- KL-divergence between two distributions X and Y
$$D_{KL}(X||Y) = \sum_{\sigma} \Pr_X[\sigma] \log (\Pr_X[\sigma]/\Pr_Y[\sigma])$$
- Equals zero iff $X = Y$; Otherwise, positive
- How to measure $D_{KL}(X||Y)$? We don't know X !
 - $D_{KL}(X||Y) = \sum_{\sigma} \Pr_X[\sigma] \log(1/\Pr_Y[\sigma]) - H(X)$
where $H(X) = -\sum_{\sigma} \Pr_X[\sigma] \log \Pr_X[\sigma]$

Perplexity (II)

- How closely does the model approximate the actual (test set) distribution?

- KL-divergence between two distributions X and Y

$$D_{KL}(X||Y) = \sum_{\sigma} \Pr_X[\sigma] \log (\Pr_X[\sigma]/\Pr_Y[\sigma])$$

- Equals zero iff $X = Y$; Otherwise, positive

- How to measure $D_{KL}(X||Y)$? We don't know X !

- $D_{KL}(X||Y) = \sum_{\sigma} \Pr_X[\sigma] \log(1/\Pr_Y[\sigma]) - H(X)$

$$\text{where } H(X) = -\sum_{\sigma} \Pr_X[\sigma] \log \Pr_X[\sigma]$$

Cross entropy
between X and Y

Perplexity (II)

- How closely does the model approximate the actual (test set) distribution?

- KL-divergence between two distributions X and Y

$$D_{KL}(X||Y) = \sum_{\sigma} \Pr_X[\sigma] \log (\Pr_X[\sigma]/\Pr_Y[\sigma])$$

- Equals zero iff $X = Y$; Otherwise, positive

- How to measure $D_{KL}(X||Y)$? We don't know X !
- $D_{KL}(X||Y) = \sum_{\sigma} \Pr_X[\sigma] \log(1/\Pr_Y[\sigma]) - H(X)$
where $H(X) = -\sum_{\sigma} \Pr_X[\sigma] \log \Pr_X[\sigma]$

Cross entropy
between X and Y

- Empirical cross entropy:

$$\frac{1}{|test|} \sum_{\sigma \in test} \log\left(\frac{1}{\Pr_Y[\sigma]}\right)$$

Perplexity vs. Empirical Cross Entropy

Perplexity vs. Empirical Cross Entropy

- Empirical Cross Entropy (ECE)

$$\frac{1}{|\#sents|} \sum_{\sigma \in test} \log\left(\frac{1}{\text{Pr}_{model}[\sigma]}\right)$$

Perplexity vs. Empirical Cross Entropy

- Empirical Cross Entropy (ECE)

$$\frac{1}{|\#sents|} \sum_{\sigma \in test} \log\left(\frac{1}{Pr_{model}[\sigma]}\right)$$

- Normalized Empirical Cross Entropy = ECE/(avg. length) =

$$\frac{1}{|\#words/\#sents|} \frac{1}{|\#sents|} \sum_{\sigma \in test} \log\left(\frac{1}{Pr_{model}[\sigma]}\right)$$
$$= \frac{1}{N} \sum_{\sigma} \log\left(\frac{1}{Pr_{model}[\sigma]}\right)$$

where $N = \#words$

Perplexity vs. Empirical Cross Entropy

- Empirical Cross Entropy (ECE)

$$\frac{1}{|\#sents|} \sum_{\sigma \in test} \log\left(\frac{1}{\text{Pr}_{model}[\sigma]}\right)$$

- Normalized Empirical Cross Entropy = ECE/(avg. length) =

$$\begin{aligned} \frac{1}{|\#words/\#sents|} \frac{1}{|\#sents|} \sum_{\sigma \in test} \log\left(\frac{1}{\text{Pr}_{model}[\sigma]}\right) \\ = \frac{1}{N} \sum_{\sigma} \log\left(\frac{1}{\text{Pr}_{model}[\sigma]}\right) \end{aligned}$$

where $N = \#words$

- How does $\frac{1}{N} \sum_{\sigma} \log\left(\frac{1}{\text{Pr}_{model}[\sigma]}\right)$ relate to perplexity?

Perplexity vs. Empirical Cross Entropy

$$\begin{aligned}\log(\text{perplexity}) &= \boxed{\frac{1}{N} \log \frac{1}{\Pr[\text{test}]}} \\ &= \frac{1}{N} \log \prod_{\sigma} \left(\frac{1}{\Pr_{\text{model}}[\sigma]} \right) \\ &= \frac{1}{N} \sum_{\sigma} \log \left(\frac{1}{\Pr_{\text{model}}[\sigma]} \right)\end{aligned}$$

Perplexity vs. Empirical Cross Entropy

$$\begin{aligned}\log_2(\text{perplexity}) &= \frac{1}{N} \log \frac{1}{\Pr[test]} \\ &= \frac{1}{N} \log \prod_{\sigma} \left(\frac{1}{\Pr_{model}[\sigma]} \right) \\ &= \frac{1}{N} \sum_{\sigma} \log \left(\frac{1}{\Pr_{model}[\sigma]} \right)\end{aligned}$$

Thus, perplexity = $\exp_{\text{Norm ECE}}^{\text{(normalized cross entropy)}}$

Perplexity vs. Empirical Cross Entropy

$$\begin{aligned}\log(\text{perplexity}) &= \frac{1}{N} \log \frac{1}{\Pr[\text{test}]} \\ &= \frac{1}{N} \log \prod_{\sigma} \left(\frac{1}{\Pr_{\text{model}}[\sigma]} \right) \\ &= \frac{1}{N} \sum_{\sigma} \log \left(\frac{1}{\Pr_{\text{model}}[\sigma]} \right)\end{aligned}$$

Thus, perplexity = \exp (normalized cross entropy)

Example perplexities for Ngram models trained on WSJ (80M words):

Unigram: 962, Bigram: 170, Trigram: 109

Introduction to smoothing of LMs

Recall example

The dog chased a cat
The cat chased away a mouse
The mouse eats cheese

What is $\text{Pr}(\text{"The dog eats cheese"})$?

Recall example

The dog chased a cat
The cat chased away a mouse
The mouse eats cheese

What is $\Pr(\text{"The dog eats cheese"})$?

$\Pr(\text{"<s> The dog eats cheese </s>"}) =$

$\Pr(\text{"The|<s>"}) \cdot \Pr(\text{"dog|The"}) \cdot \Pr(\text{"eats|dog"}) \cdot \Pr(\text{"cheese|eats"}) \cdot \Pr(\text{"</s>|cheese"}) =$

$3/3 \cdot 1/3 \cdot 0/1 \cdot 1/1 \cdot 1/1 = 0!$

Recall example

The dog chased a cat
The cat chased away a mouse
The mouse eats cheese

What is $\Pr(\text{"The dog eats cheese"})$?

$\Pr(\text{"<s> The dog eats cheese </s>"}) =$

$\Pr(\text{"The|<s>"}) \cdot \Pr(\text{"dog|The"}) \cdot \Pr(\text{"eats|dog"}) \cdot \Pr(\text{"cheese|eats"}) \cdot \Pr(\text{"</s>|cheese"}) =$

$\frac{3}{3} \cdot \frac{1}{3} \cdot \frac{0}{1} \cdot \frac{1}{1} \cdot \frac{1}{1} = 0!$ Due to unseen bigrams

Unseen Ngrams

Unseen Ngrams

- Even with MLE estimates based on counts from large text corpora, there will be many unseen bigrams/trigrams that never appear in the corpus

Unseen Ngrams

- Even with MLE estimates based on counts from large text corpora, there will be many unseen bigrams/trigrams that never appear in the corpus
- If any unseen Ngram appears in a test sentence, the sentence will be assigned probability 0

Unseen Ngrams

- Even with MLE estimates based on counts from large text corpora, there will be many unseen bigrams/trigrams that never appear in the corpus
- If any unseen Ngram appears in a test sentence, the sentence will be assigned probability 0
- Problem with MLE estimates: maximises the likelihood of the observed data by assuming anything unseen cannot happen and overfits to the training data
- **Smoothing methods:** Reserve some probability mass to Ngrams that don't occur in the training corpus

Add-one (Laplace) smoothing

Simple idea: Add one to all bigram counts. That means,

$$\Pr_{ML}(w_i|w_{i-1}) = \frac{\pi(w_{i-1}, w_i)}{\pi(w_{i-1})}$$

becomes

$$\Pr_{Lap}(w_i|w_{i-1}) = \frac{\pi(w_{i-1}, w_i) + 1}{\pi(w_{i-1}) + V} \quad \checkmark$$

where V is the vocabulary size

Example: Bigram counts

*No
smoothing*

	i	<u>want</u>	to	eat	chinese	food	lunch	spend
<u>i</u>	5	<u>827</u>	0	9	0	0	0	2
want	2	0	<u>608</u>	1	6	6	5	1
to	2	0	4	<u>686</u>	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

*Laplace
(Add-one)
smoothing*

	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

Example: Bigram probabilities

**No
smoothing**

	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

**Laplace
(Add-one)
smoothing**

	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

Example: Bigram probabilities

**No
smoothing**

	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

**Laplace
(Add-one)
smoothing**

	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

Example: Bigram probabilities

**No
smoothing**

	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

**Laplace
(Add-one)
smoothing**

	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

Laplace smoothing moves too much probability mass to unseen events!

Add- α Smoothing

Instead of 1, add $\alpha < 1$ to each count

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

Add- α Smoothing

Instead of 1, add $\alpha < 1$ to each count

$$\text{Pr}_\alpha(w_i|w_{i-1}) = \frac{\pi(w_{i-1}, w_i) + \alpha}{\pi(w_{i-1}) + \alpha V}$$

Choosing α :

- Train model on training set using different values of α
- Choose the value of α that minimizes cross entropy on the development set

Smoothing or discounting

- Smoothing can be viewed as **discounting** (lowering) some probability mass from seen Ngrams and redistributing discounted mass to unseen events
- i.e. probability of a bigram with Laplace smoothing

$$\Pr_{Lap}(w_i|w_{i-1}) = \frac{\pi(w_{i-1}, w_i) + 1}{\pi(w_{i-1}) + V}$$

- can be written as

$$\Pr_{Lap}(w_i|w_{i-1}) = \frac{\pi^*(w_{i-1}, w_i)}{\pi(w_{i-1})}$$

- where discounted count $\pi^*(w_{i-1}, w_i) = (\pi(w_{i-1}, w_i) + 1) \frac{\pi(w_{i-1})}{\pi(w_{i-1}) + V}$

Example: Bigram adjusted counts

*No
smoothing*

	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

*Laplace
(Add-one)
smoothing*

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Smoothing or discounting

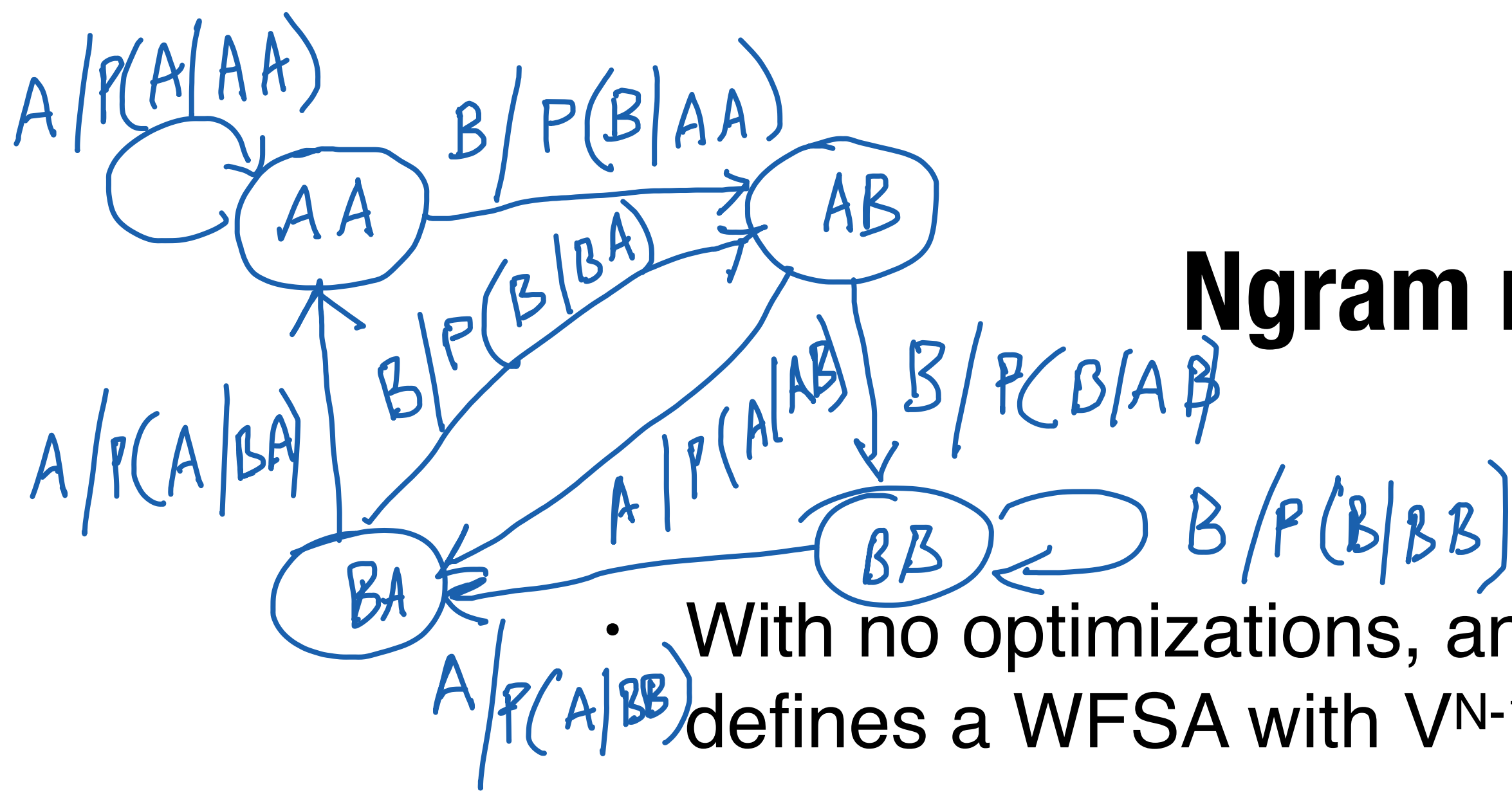
- Smoothing can be viewed as **discounting** (lowering) some probability mass from seen Ngrams and redistributing discounted mass to unseen events
- i.e. probability of a bigram with Laplace smoothing

$$\Pr_{Lap}(w_i|w_{i-1}) = \frac{\pi(w_{i-1}, w_i) + 1}{\pi(w_{i-1}) + V}$$

- can be written as

$$\Pr_{Lap}(w_i|w_{i-1}) = \frac{\pi^*(w_{i-1}, w_i)}{\pi(w_{i-1})}$$

- where discounted count $\pi^*(w_{i-1}, w_i) = (\pi(w_{i-1}, w_i) + 1) \frac{\pi(w_{i-1})}{\pi(w_{i-1}) + V}$
- More sophisticated smoothing/discounting schemes in the next lecture!



- With no optimizations, an Ngram over a vocabulary of V words defines a WFSA with V^{N-1} states and V^N edges.

- Example: Consider a trigram model for a two-word vocabulary, A B.
- 4 states representing bigram histories, A_A, A_B, B_A, B_B
- 8 arcs transitioning between these states
- Clearly not practical when V is large.
- Resort to backoff language models (More in the next lecture)