

UMass · CS685 | Advanced Natural Language Processing (2020)

CS685 (2020) · 课程资料包 @ShowMeAI



视频

中英双语字幕

课件

一键打包下载

笔记

官方笔记翻译

代码

作业项目解析



视频 · B 站 [扫码或点击链接]

<https://www.bilibili.com/video/BV1BL411t7RV>



课件 & 代码 · 博客 [扫码或点击链接]

<http://blog.showmeai.tech/umass-cs685>

NLP

迁移学习

语言模型 问答系统 文本生成 BERT

语义解析

知识推理

模型蒸馏

transformer

GPT-3

注意力机制

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



微信公众号

资料下载方式 2：扫码点击**底部菜单栏**

称为 **AI 内容创作者**？回复 [添砖加瓦]

language modeling

CS 685, Fall 2020

Introduction to Natural Language Processing

<http://people.cs.umass.edu/~miyyer/cs685/>

Mohit Iyyer

College of Information and Computer Sciences

University of Massachusetts Amherst

some slides from Dan Jurafsky and Richard Socher

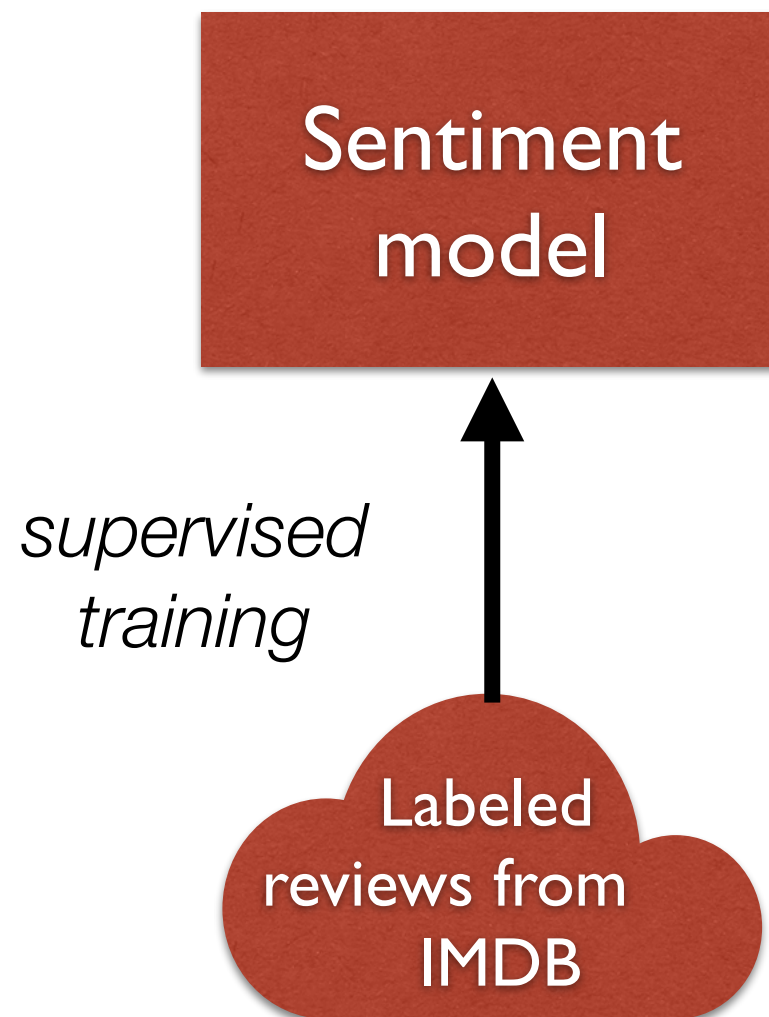
questions from last time...

- Cheating concerns on exam?
 - we're still thinking about ways to mitigate this
- Final project group size?
 - Will be 4 with few exceptions
 - Please use Piazza to form teams by 9/4 (otherwise we will randomly assign you)
- HW0?
 - Out today, due 9/4. Start early especially if you have a limited coding / math background!

Let's say I want to train a model for *sentiment analysis*

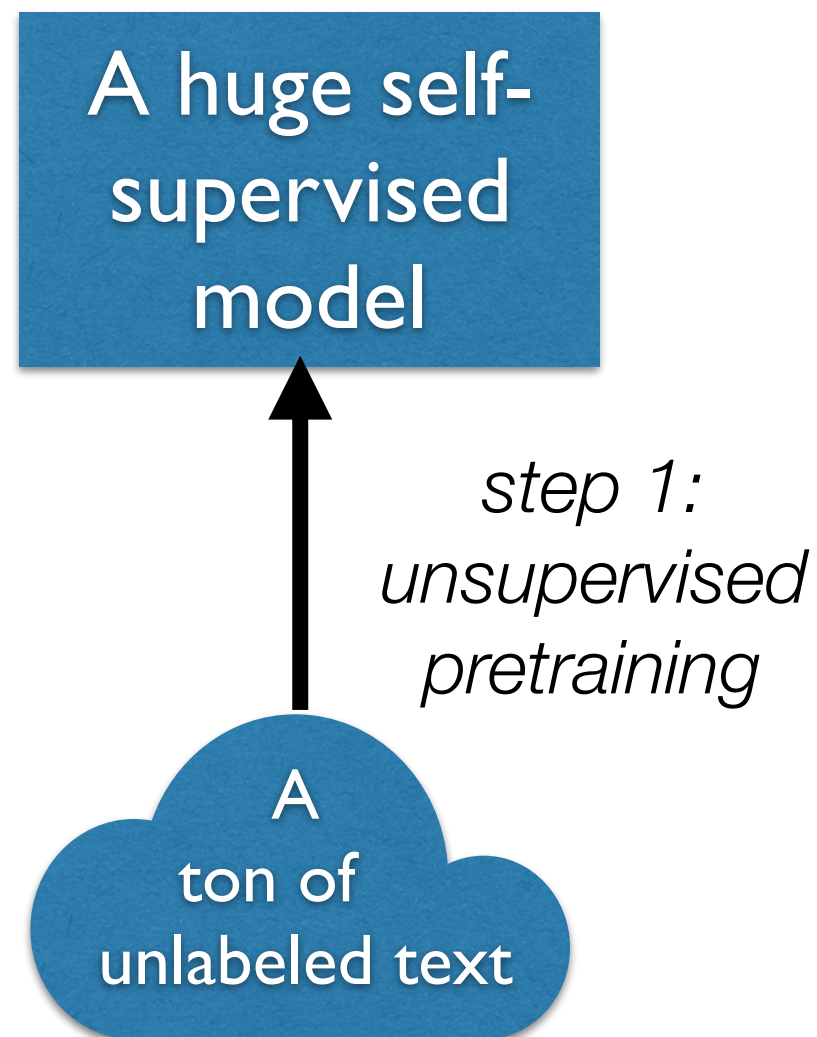
Let's say I want to train a model for *sentiment analysis*

In the past, I would simply train a *supervised* model on labeled sentiment examples (i.e., review text / score pairs from IMDB)



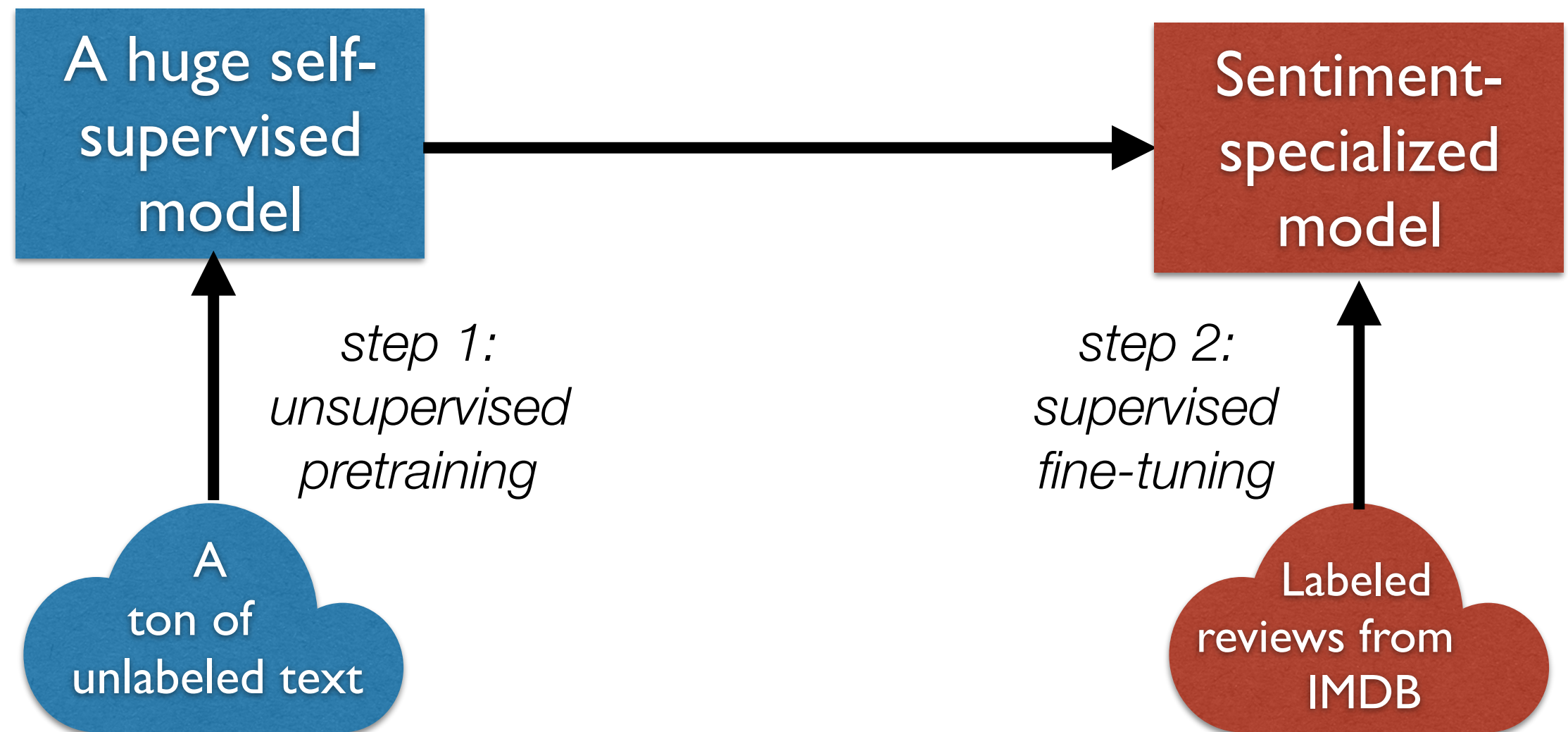
Let's say I want to train a model for *sentiment analysis*

Nowadays, however, we use *transfer learning*:

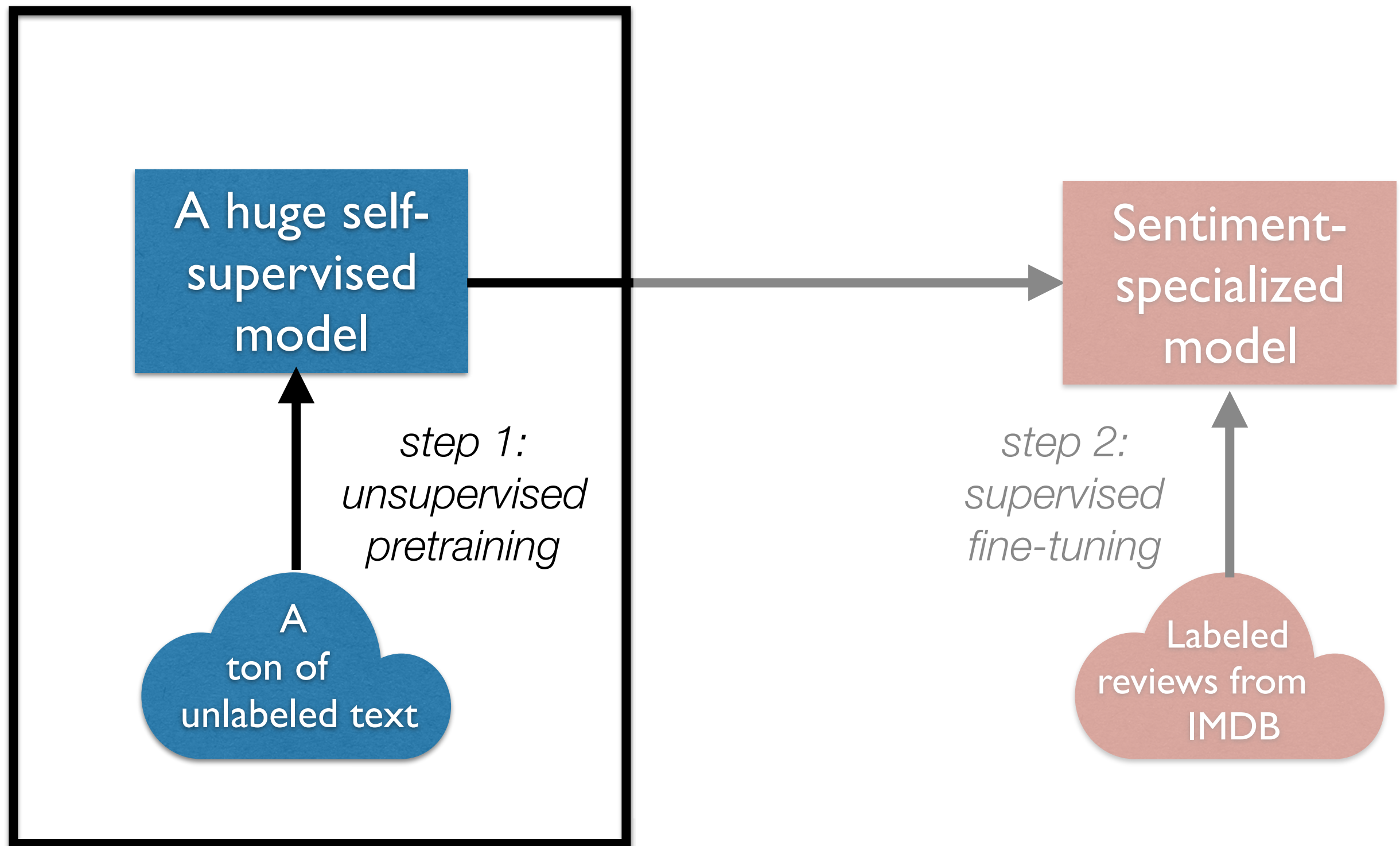


Let's say I want to train a model for *sentiment analysis*

Nowadays, however, we use *transfer learning*:



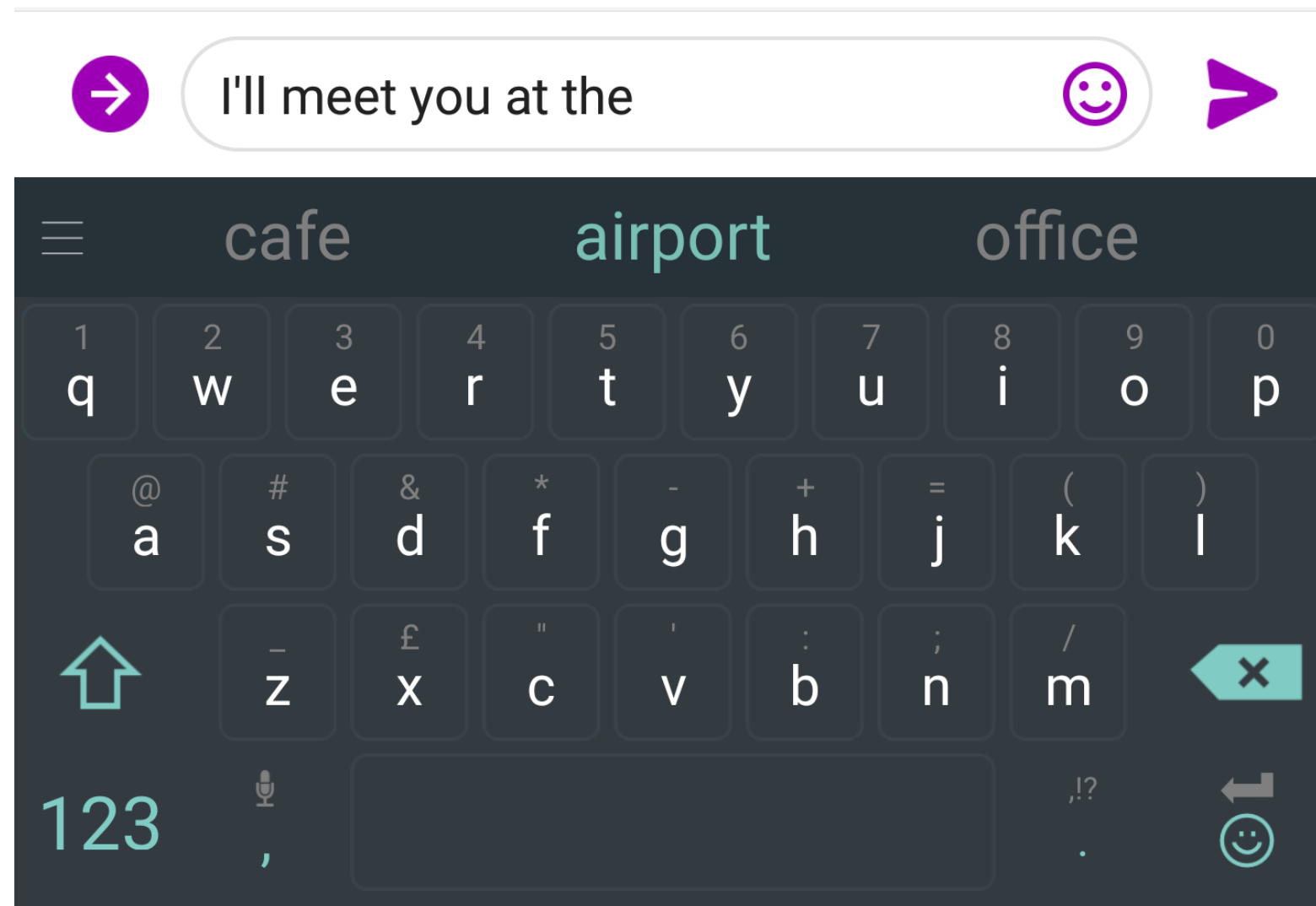
This lecture: **language modeling**, which forms the core of most self-supervised NLP approaches



Language models assign a probability to a piece of text


- why would we ever want to do this?
- translation:
 - $P(\text{i flew to the movies}) \lll P(\text{i went to the movies})$
- speech recognition:
 - $P(\text{i saw a van}) \ggg P(\text{eyes awe of an})$

You use Language Models every day!



You use Language Models every day!



what is the | 

what is the **weather**
what is the **meaning of life**
what is the **dark web**
what is the **xfl**
what is the **doomsday clock**
what is the **weather today**
what is the **keto diet**
what is the **american dream**
what is the **speed of light**
what is the **bill of rights**

[Google Search](#) [I'm Feeling Lucky](#)

Probabilistic Language Modeling

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

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

- Related task: probability of an upcoming word:

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

- A model that computes either of these:

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

How to compute $P(W)$

- How to compute this joint probability:
 - $P(\text{its, water, is, so, transparent, that})$
- Intuition: let's rely on the Chain Rule of Probability

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

$$P(B | A) = P(A,B)/P(A) \quad \text{Rewriting: } P(A,B) = P(A)P(B | A)$$

- More variables:

$$P(A,B,C,D) = P(A)P(B | A)P(C | A,B)P(D | A,B,C)$$

- The Chain Rule in General

$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2 | x_1)P(x_3 | x_1, x_2) \dots P(x_n | x_1, \dots, x_{n-1})$$

The Chain Rule applied to compute joint probability of words in sentence

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

$$\begin{aligned} P(\text{"its water is so transparent"}) = & \\ & P(\text{its}) \times P(\text{water} | \text{its}) \times P(\text{is} | \text{its water}) \\ & \times P(\text{so} | \text{its water is}) \times P(\text{transparent} | \text{its water is so}) \end{aligned}$$

The Chain Rule applied to compute joint probability of words in sent

In HWO, we refer to this as a “prefix”

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

$P(\text{“its water is so transparent”}) =$
 $P(\text{its}) \times P(\text{water} | \text{its}) \times P(\text{is} | \text{its water})$
 $\times P(\text{so} | \text{its water is}) \times P(\text{transparent} | \text{its water is so})$

How to estimate these probabilities

- Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{\textit{Count}(\text{its water is so transparent that the})}{\textit{Count}(\text{its water is so transparent that})}$$

How to estimate these probabilities

- Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{\textit{Count}(\text{its water is so transparent that the})}{\textit{Count}(\text{its water is so transparent that})}$$

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption

- Simplifying assumption:



Andrei Markov (1856~1922)

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

- Or maybe

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

Markov Assumption

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

- In other words, we approximate each component in the product

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

Simplest case: Unigram model

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

Some automatically generated sentences from a unigram model:

fifth, an, of, futures, the, an, incorporated, a, a,
the, inflation, most, dollars, quarter, in, is, mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the

How can we generate text
from a language model?

Approximating Shakespeare

1
gram

–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have

–Hill he late speaks; or! a more to leg less first you enter

2
gram

–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.

–What means, sir. I confess she? then all sorts, he is trim, captain.

3
gram

–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

–This shall forbid it should be branded, if renown made it empty.

4
gram

–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;

–It cannot be but so.

N-gram models

- We can extend to trigrams, 4-grams, 5-grams
- 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.”
- But we can often get away with N-gram models

In the next video, we will look at some models that can theoretically handle some of these longer-term dependencies

Estimating bigram probabilities

- The Maximum Likelihood Estimate (MLE)
 - relative frequency based on the empirical counts on a training set

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

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

c — count

An example

$$P(w_i | w_{i-1}) \stackrel{\text{MLE}}{=} \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

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

$$P(\text{I} | \text{<s>}) = \frac{2}{3} = .67 \qquad P(\text{Sam} | \text{<s>}) = ???$$

$$P(\text{</s>} | \text{Sam}) = \frac{1}{2} = 0.5 \qquad P(\text{Sam} | \text{am}) = ???$$

An example

$$P(w_i | w_{i-1}) \stackrel{\text{MLE}}{=} \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

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

$$P(\text{I} | \text{<s>}) = \frac{2}{3} = .67 \quad P(\text{Sam} | \text{<s>}) = \frac{1}{3} = .33 \quad P(\text{am} | \text{I}) = \frac{2}{3} = .67$$

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

Important terminology: a word **type** is a unique word in our vocabulary, while a **token** is an occurrence of a word type in a dataset.

An example

$$P(w_i | w_{i-1}) \stackrel{\text{MLE}}{=} \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

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

$$\begin{aligned}
 P(\text{I} | \text{<s>}) &= \frac{2}{3} = .67 & P(\text{Sam} | \text{<s>}) &= \frac{1}{3} = .33 & P(\text{am} | \text{I}) &= \frac{2}{3} = .67 \\
 P(\text{</s>} | \text{Sam}) &= \frac{1}{2} = 0.5 & P(\text{Sam} | \text{am}) &= \frac{1}{2} = .5 & P(\text{do} | \text{I}) &= \frac{1}{3} = .33
 \end{aligned}$$

A bigger example: Berkeley Restaurant Project 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

note: this is only a subset of the (much bigger) bigram count table

- 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

Raw bigram probabilities $P(w_i | w_{i-1}) \stackrel{\text{MLE}}{=} \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$

- Normalize by unigrams:

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

- Result:

	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

Bigram estimates of sentence probabilities

$$P(<s> \text{ I want english food } </s>) =$$

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

$$\times P(\text{want} | \text{I})$$

$$\times P(\text{english} | \text{want})$$

$$\times P(\text{food} | \text{english})$$

$$\times P(</s> | \text{food})$$

$$= .000031$$

these probabilities get super tiny when we have longer inputs w/ more infrequent words... how can we get around this?

logs to avoid underflow

$$\log \prod p(w_i | w_{i-1}) = \sum \log p(w_i | w_{i-1})$$

Example with unigram model on a sentiment dataset:

logs to avoid underflow

$$\log \prod p(w_i | w_{i-1}) = \sum \log p(w_i | w_{i-1})$$


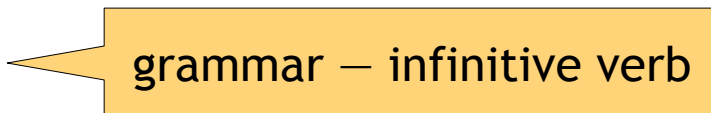
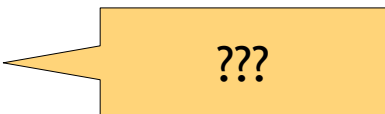
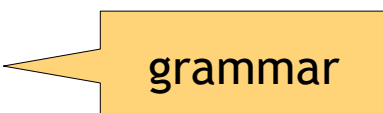
Example with unigram model on a sentiment dataset:

$$p(i) \cdot p(\text{love})^5 \cdot p(\text{the}) \cdot p(\text{movie}) = 5.95374181\text{e-}7$$

$$\log p(i) + 5 \log p(\text{love}) + \log p(\text{the}) + \log p(\text{movie})$$

$$= -14.3340757538$$

What kinds of knowledge?

- $P(\text{english} | \text{want}) = .0011$  about the world
- $P(\text{chinese} | \text{want}) = .0065$
- $P(\text{to} | \text{want}) = .66$  grammar — infinitive verb
- $P(\text{eat} | \text{to}) = .28$
- $P(\text{food} | \text{to}) = 0$  ???
- $P(\text{want} | \text{spend}) = 0$  grammar
- $P(i | \langle s \rangle) = .25$

Language Modeling Toolkits

- SRILM

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

- KenLM

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

Evaluation: How good is our model?

- Does our language model prefer good sentences to bad ones?
 - Assign higher probability to “real” or “frequently observed” sentences
 - Than “ungrammatical” or “rarely observed” sentences?
- We train parameters of our model on a **training set**.
- We test the model’s performance on data we haven’t seen.
 - A **test set** is an unseen dataset that is different from our training set, totally unused.
 - An **evaluation metric** tells us how well our model does on the test set.

Evaluation: How good is our model?

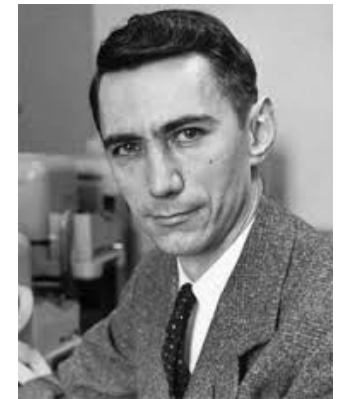
- The goal isn't to pound out fake sentences!
 - Obviously, generated sentences get “better” as we increase the model order
 - More precisely: using maximum likelihood estimators, higher order is always better likelihood on **training set**, but not **test set**

Training on the test set

- We can't allow test sentences into the training set
- We will assign it an artificially high probability when we set it in the test set
- “Training on the test set”
- Bad science!

This advice is generally applicable to any downstream task! Do NOT do this in your final projects unless you want to lose a lot of points :)

Intuition of Perplexity



Claude Shannon
(1916~2001)

- The Shannon Game:

- How well can we predict the next word?

I always order pizza with cheese and _____

The 33rd President of the US was _____

I saw a _____

- Unigrams are terrible at this game. (Why?)
- A better model of a text
 - is one which assigns a higher probability to the word that actually occurs
 - compute per word log likelihood
(M words, m test sentence s_i)

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

....

fried rice 0.0001

....

and 1e-100

$$l = \frac{1}{M} \sum_{i=1}^m \log p(s_i)$$

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest $P(\text{sentence})$

Perplexity is the inverse probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

Chain rule:

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

For bigrams:

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

Minimizing perplexity is the same as maximizing probability

Perplexity as branching factor

Let's suppose a sentence consisting of random digits

What is the perplexity of this sentence according to a model that assign $P=1/10$ to each digit?

$$\begin{aligned} \text{PP}(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}\right)^{-1} \\ &= 10 \end{aligned}$$

Lower perplexity = better model

- Training 38 million words, test 1.5 million words, Wall Street Journal

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Zero probability bigrams

- Bigrams with zero probability
 - mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can't divide by 0)!

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$
$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

for bigram

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

Q: How do we deal with ngrams of zero probabilities?

Shakespeare as corpus

- $N=884,647$ tokens, $V=29,066$
- Shakespeare produced 300,000 bigram types out of $V^2= 844$ million possible bigrams.
 - So 99.96% of the possible bigrams were never seen (have zero entries in the table)

Zeros

Training set:

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

- Test set
 - ... denied the offer
 - ... denied the loan

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

The intuition of smoothing (from Dan Klein)

- When we have sparse statistics:

$P(w \mid \text{denied the})$

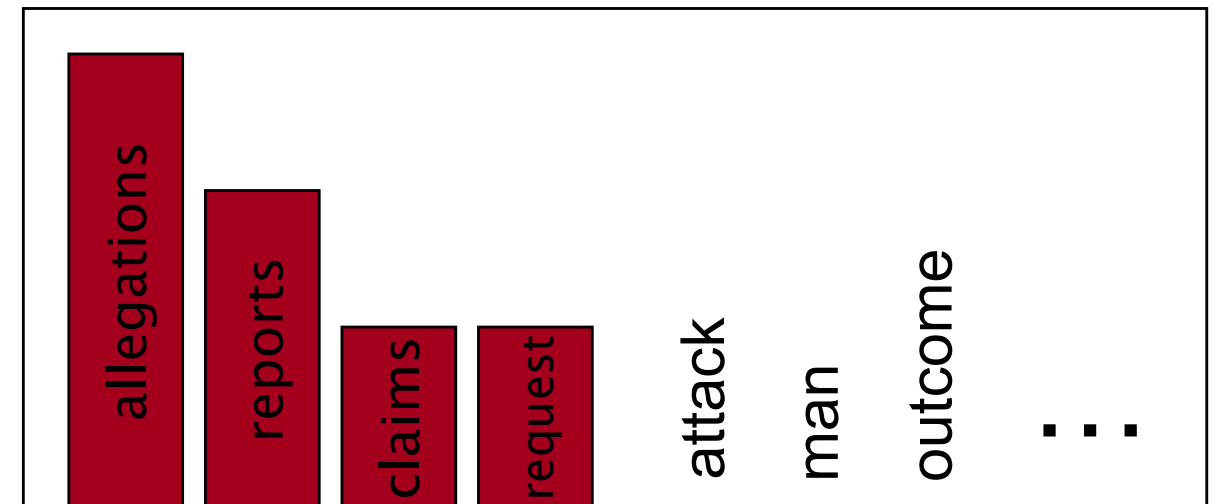
3 allegations

2 reports

1 claims

1 request

7 total



- Steal probability mass to generalize better

$P(w \mid \text{denied the})$

2.5 allegations

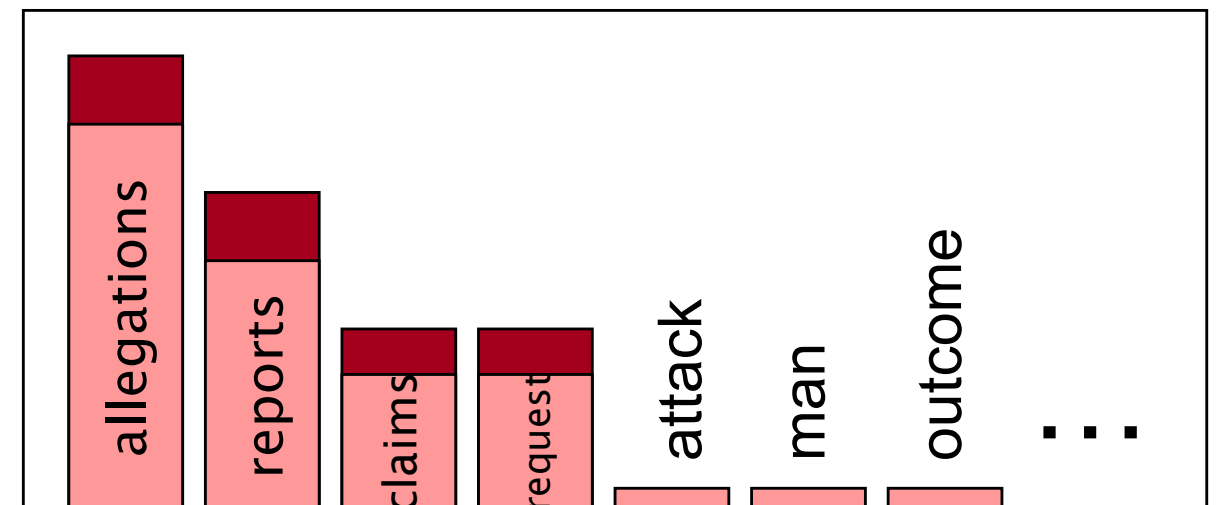
1.5 reports

0.5 claims

0.5 request

2 other

7 total



Be on the look out for...

- HW0, out today and due 9/4 on Gradescope!
- Lectures on neural language modeling and backpropagation, coming next Monday / Wednesday!
- Please use Piazza to form final project teams!

UMass · CS685 | Advanced Natural Language Processing (2020)

CS685 (2020) · 课程资料包 @ShowMeAI



视频

中英双语字幕



课件

一键打包下载



笔记

官方笔记翻译



代码

作业项目解析



视频 · B 站 [扫码或点击链接]

<https://www.bilibili.com/video/BV1BL411t7RV>



课件 & 代码 · 博客 [扫码或点击链接]

<http://blog.showmeai.tech/umass-cs685>

NLP

迁移学习

语言模型 问答系统 文本生成 BERT

语义解析

知识推理

模型蒸馏

transformer

GPT-3

注意力机制

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



微信公众号

资料下载方式 2：扫码点击**底部菜单栏**

称为 **AI 内容创作者**？回复 [添砖加瓦]