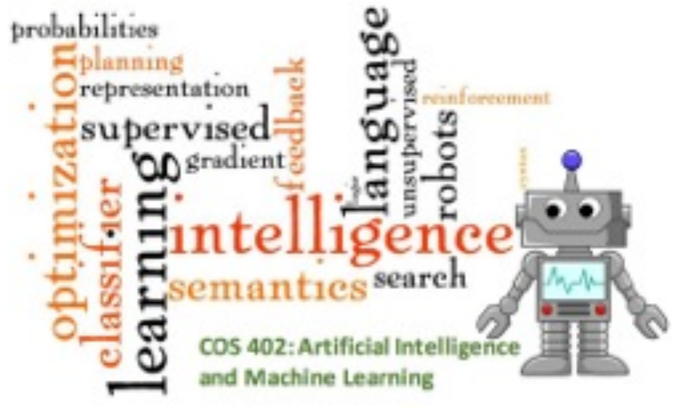


COS 402 – Machine Learning and Artificial Intelligence Fall 2016

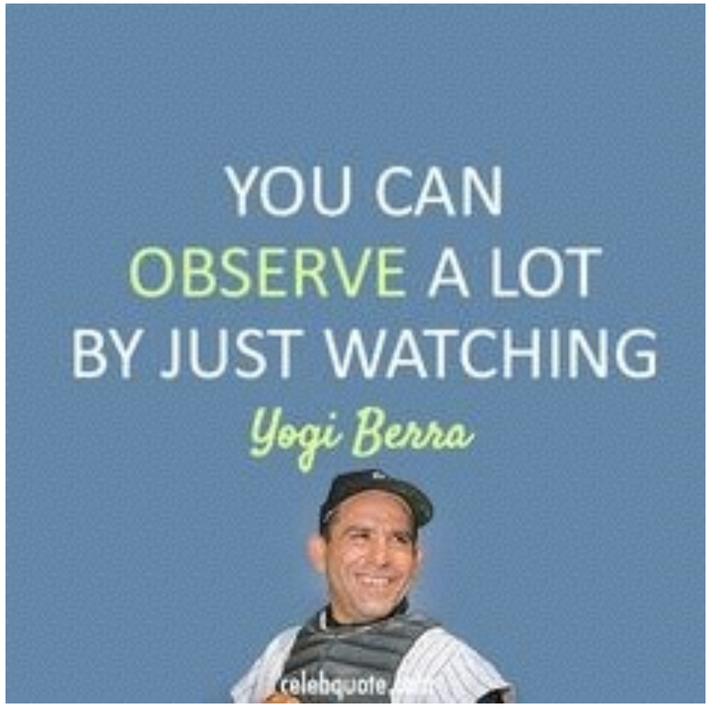
Lecture 9: Language models (n-grams)

Sanjeev Arora Elad Hazan

PRINCETON UNIVERSITY

(Borrows from slides of D. Jurafsky Stanford U.)

Thus far in the course: Learning from labeled examples. (SUPERVISED LEARNING
Today: Learning from lots of unlabeled/unannotated data. (UNSUPERVISED LEARN

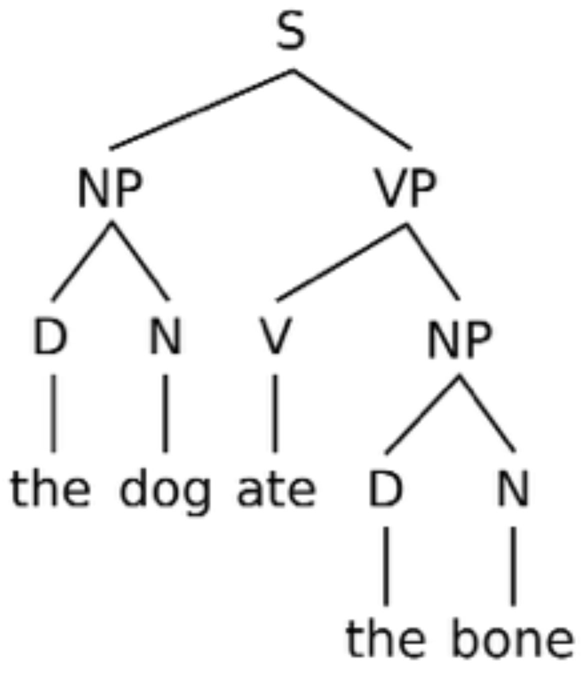


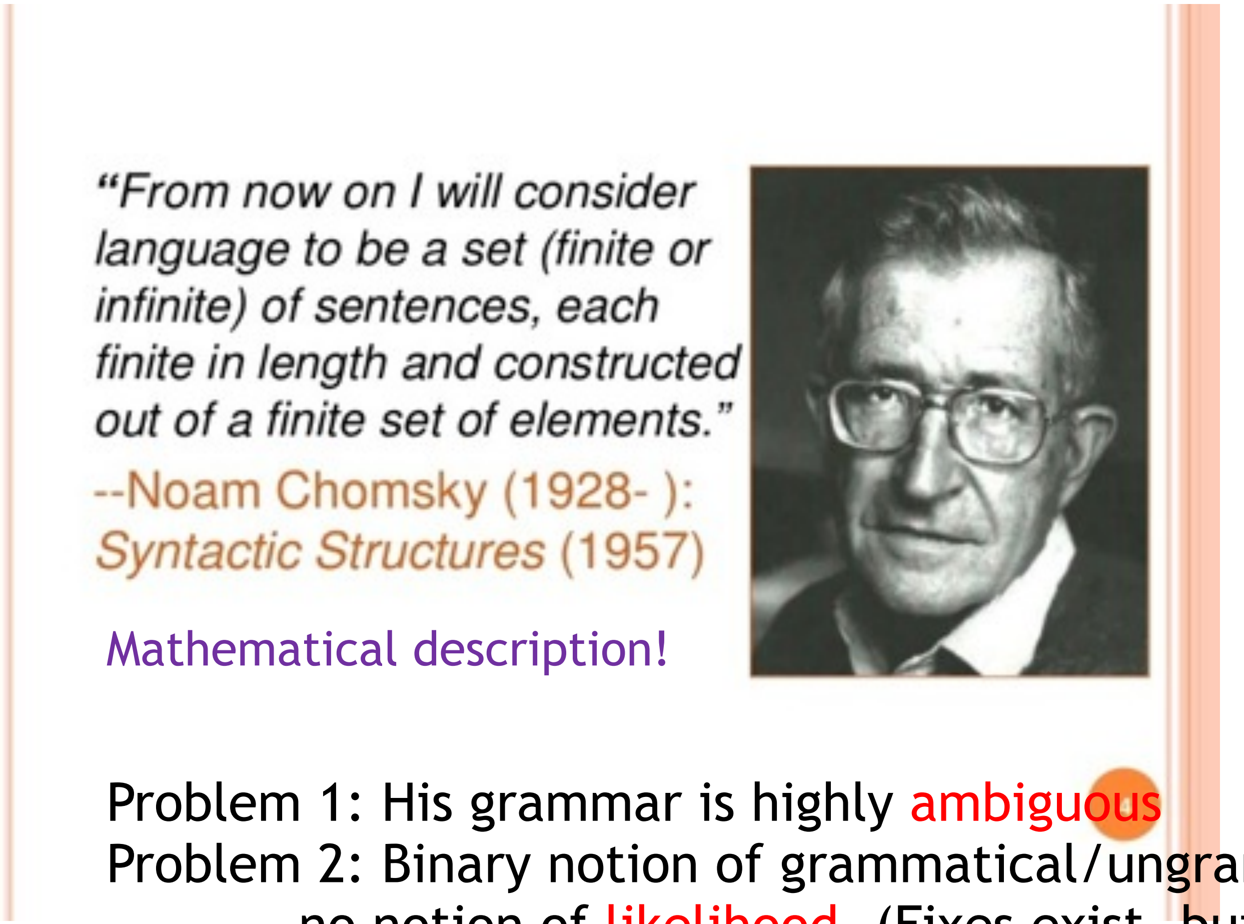
e.g., to understand structure of language,
computer can try to go through a big text corpus.

(Will also continue with this theme next time.)

What can you learn by just reading?

1) Syntax





"From now on I will consider language to be a set (finite or infinite) of sentences, each finite in length and constructed out of a finite set of elements."

--Noam Chomsky (1928-): *Syntactic Structures* (1957)

Mathematical description!

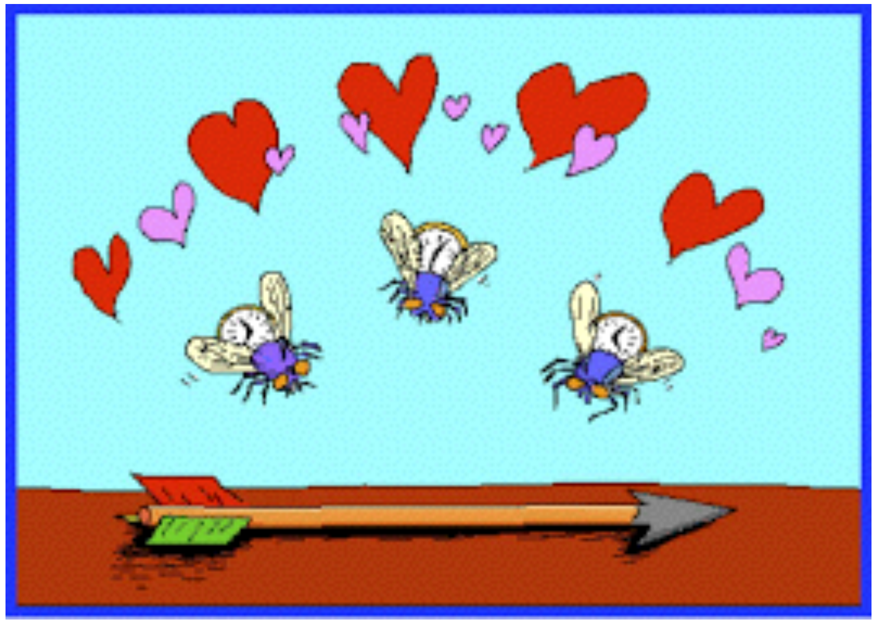
Problem 1: His grammar is highly ambiguous

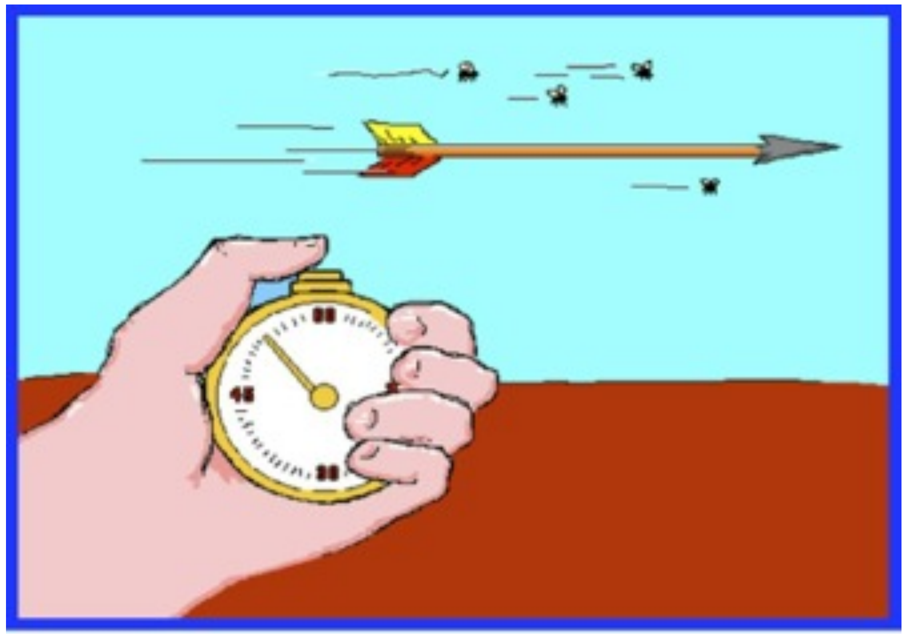
Problem 2: Binary notion of grammatical/ungrammatrical; no notion of likelihood. (Fixes exist, but..)

Possible to learn basic grammar from reading alone; won't do today.

Ambiguity: multiple syntactic interpretations of same sentence

"Time flies like an arrow."





Several other parsings; try to find a few… !!

Figure credit: Bill DeSmedt

Ambiguities of all kinds are a fact of life in computational linguistics; won't study in this course.

This lecture:
Simple, even naïve approach to language modeling

Probabilistic model of language

- Assigns a probability to every word sequence (grammatical or not)
  $P[w_1\ w_2\ w_3\ \ldots\ w_n]$

Related: $P(w_5 | w_1, w_2, w_3)$ "conditional probability of w_5 after w_1, w_2, w_3 "

Typical Use: Improve other language processing tasks:

- Speech recognition
  "I ate a cherry" is a more likely sentence than "Eye eightuh Jerry"
- Machine translation.
  Pr[high winds tonight] > Pr[large winds tonight]
- Context sensitive spelling correctors
  "Their are problems wit this sentence."
- Sentence completion
  "Please turn off your …."

Philosophical debate

Chomsky

Ludicrous to think of "Probability of a sentence."

"Colorless green ideas sleep furiously."

Peter Norvig

"Sentence probability" is clearly culture and speaker dependent. But is approximate way to measure likelihood of usage. Useful!

This sentence had nonzero probability, as shown by the fact that you uttered it.

Bayes Rule/Chain Rule

- Recall definition of conditional probabilities

  $P(AB) = P(A)\,P(B|A)$

- More variables:

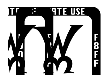
  $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$

- The Chain Rule in General

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

Example of applying Chain Rule

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



*P(*You like green cream*)* = *P*(You)*P(*like | You) *P(*green|You like) *P(*cream|You like green)

Actually estimating probabilities from a text corpus

*P(*You like green cream*)* = *P*(You) *P(*like | You) *P(*green|You like) *P(*cream|You like green)

How can we estimate these probabilities empirically?

$$P(\text{You}) = \frac{Count(\text{“You”})}{Total\ \#\ of\ words}$$

$$P(\text{like} \mid \text{You}) = \frac{Count(\text{“You like”})}{Count(\text{“You”})}$$

$$P(\text{cream} \mid \text{You like green}) = \frac{Count(\text{“You like green cream”})}{Count(\text{“You like green”})}$$

Not enough data!!

Sparse data problem

V = # of words in Oxford English dictionary = 2×10^6

Number of pairs of words = $V^2 = 4 \times 10^{12}$ (4 Trillion)

Number of triples of words = $V^3 = 8 \times 10^{18}$ (exceeds worldwide data storage)

Neither enough data nor storage to train the language model we desire.

Must settle for approximation!

Simplest approximation: unigram

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

P(“You like green cream”)≈ P(“You”)P(“like”)P(“green”)P(“cream”)

Overestimates probability for this rare sentence since all words in it are fairly common.
(remember Chomsky’s critique!)

Bigram Model: Prob of next word depends only on last word.

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

This is small, which lowers probability of this sentence. (But still too high.)

P("You like green cream")≈
P("You")P("like"|"You")P("green"|"like")P("cream"|"green")

Contrast with full chain rule:
*P(*You like green cream*)* = *P*(You)*P(*like | You) *P(*green|You like) *P(*cream|You like gree

Insufficient data to even train the full bigram m
V² is too large. Let's return to this later.

Jargon map

Unigram: single word
Bigram: Adjacent pair of words
Trigram: Adjacent Triple of words

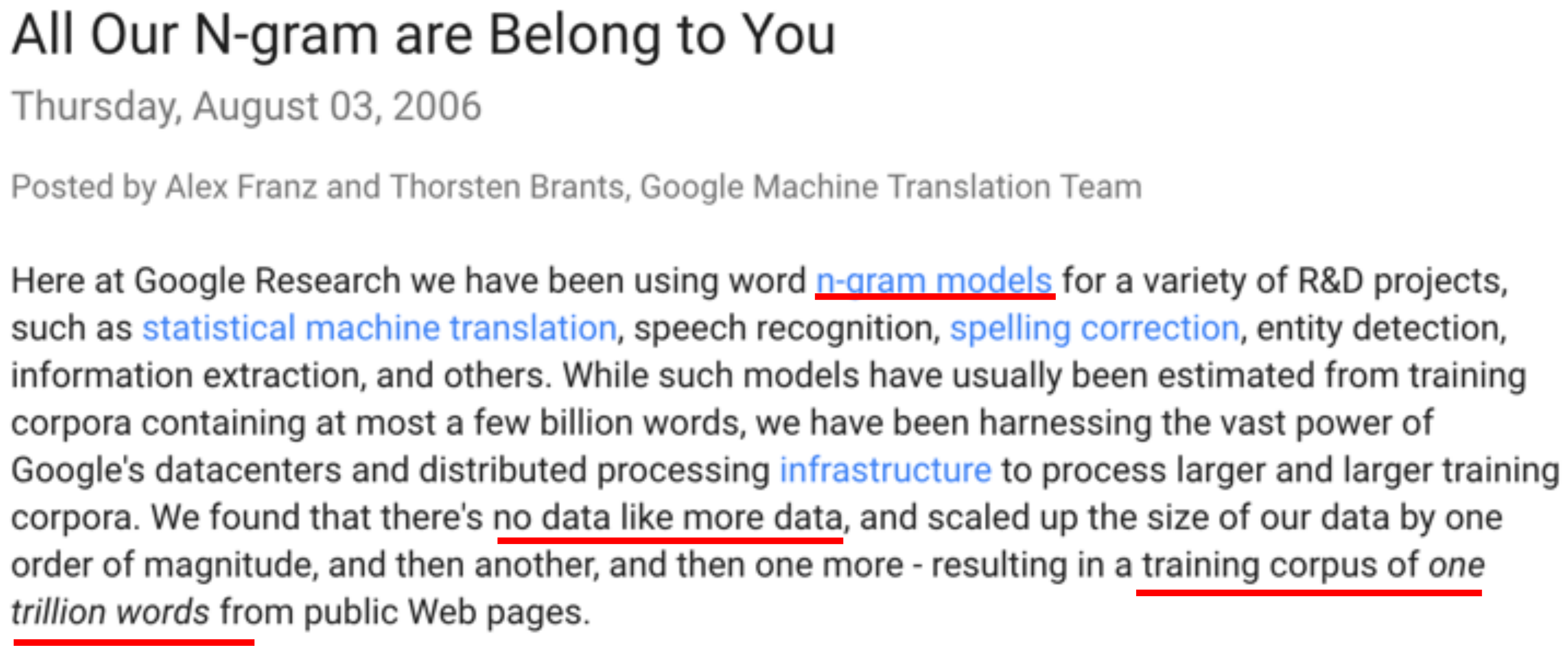
Sanity check:

Number of bigrams in a sentence of length N? N-1

Number of trigrams in a sentence of length N? N-2

Google n-gram dataset

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html



All Our N-gram are Belong to You

Thursday, August 03, 2006

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects, such as statistical machine translation, speech recognition, spelling correction, entity detection, information extraction, and others. While such models have usually been estimated from training corpora containing at most a few billion words, we have been harnessing the vast power of Google's datacenters and distributed processing infrastructure to process larger and larger training corpora. We found that there's no data like more data, and scaled up the size of our data by one order of magnitude, and then another, and then one more - resulting in a training corpus of *one trillion words* from public Web pages.

(Play with n-grams data at https://books.google.com/ngrams)

Aside: Markovian Models

Sequential models for with limited memory of past.

K-th order model : Prob. of next symbol depends
only upon last K-symbols.
(rough definition!)

A. A. Markov

K=0 ➔ Unigram; K =1 ➔ Bigram.

(Chomsky): Language is not markovian; long-range dependencies.
(i.e., no finite K suffices)

”Bulldogs Bulldogs Bulldogs Fight Fight Fight”!

(Get it? E.g., Bulldogs that bulldogs fight, fight.)

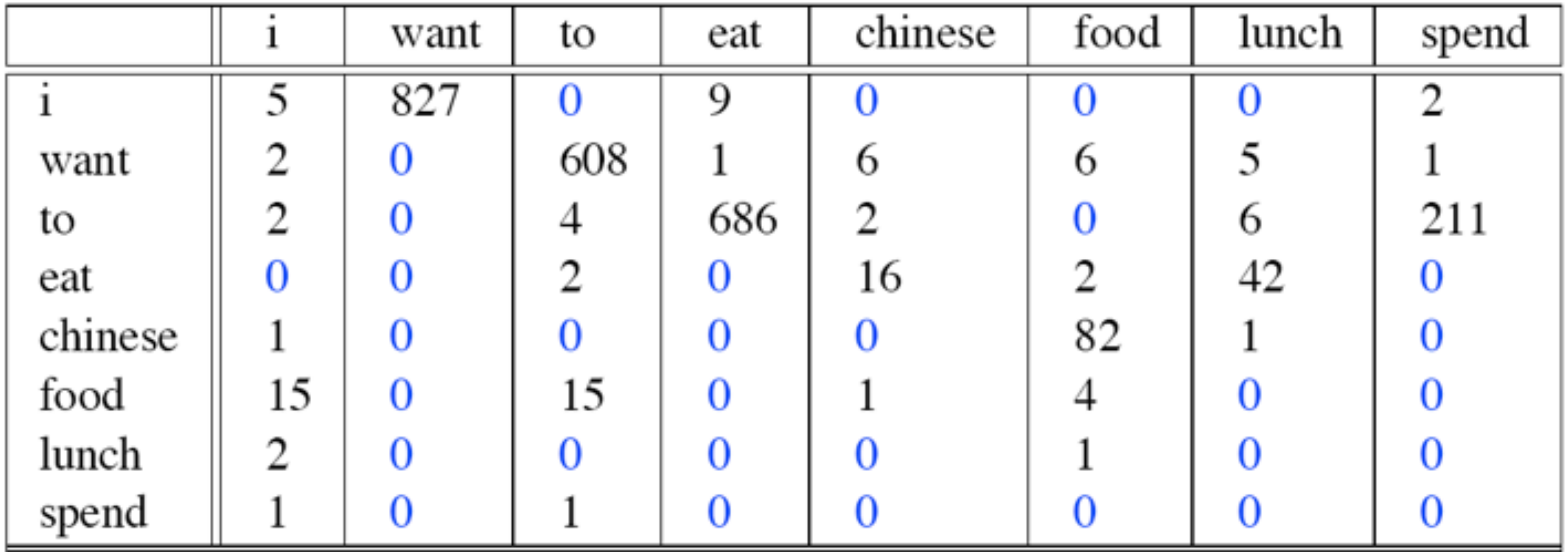
Next few slides: A worked-out example from D. Jurafsky, Stanford

(data from Berkeley Restaurant Project)

Berkeley restaurant project: Sample queries from users

- 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



	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

Converting to probabilities

Estimation rule $$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

Bigram

Unigram

Unigram frequencies

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

Implementation note for n-gram models

Customary to pad each sentence with ”start” <s> and “end” </s> symbols

You like green cream → <s> You like green cream </s>

Reasons:

(Theoretical) Ensures that sum of probabilities of all (infinitely many) sentences i

Optional: Convince yourself of this.

(Practical): Allows us to talk about the probability that say “You” starts a sentence

(Unigram Pr(You) does not capture this.)

Estimation of probability of sentence

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

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

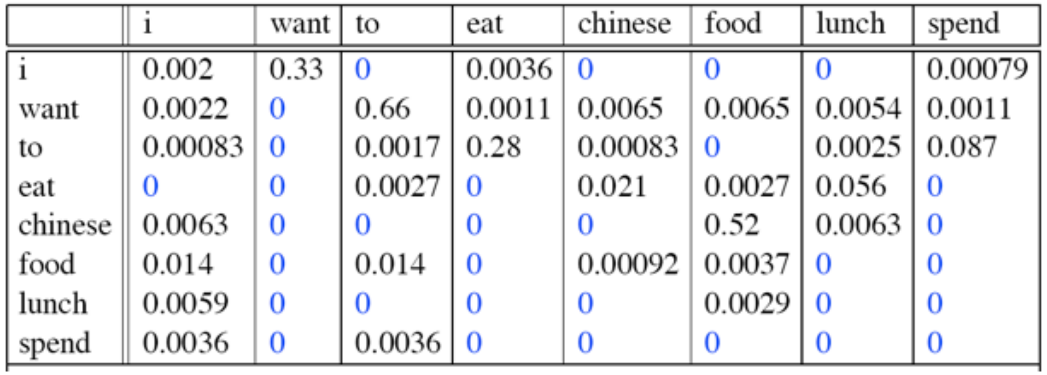
$\times\ P(\text{want}|\text{I})$

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

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

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

$= .000031$



	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

NB: Do everything in log space

- Avoid underflow
- (also adding is faster than multiplying)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

“Knowledge” captured by model parameters

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

Next few slides: How to evaluate language models (“Perplexity”)

Evaluating Language Models

- Does our language model give higher probability to good sentences ("real" or "frequently observed") compared to bad ones?
- Methology:
  (a) Train model on a **training set**.
  (b) Test model's performance on previously unseen data **(test set)**
  (c) Have **evaluation metric** to quantify how well our model does on the test set.

Note: Analogous to methology for supervised learning

Popular evaluation metric: Perplexity score given by the model to **test**

Perplexity: Intuition

- 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 ____
  - Unigram models terrible at this game. (Why?)

mushrooms	0.1
pepperoni	0.1
anchovies	0.01
….	
fried rice	0.0001
….	
and	1e-100

Model is better if in the **test data** it assigns higher probability to word that *actually* occurs.

Perplexity definition

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

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

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

(NB: Minimize Perplexity → Maximize probability of the sentence.)

Chain rule:

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

For bigrams:

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

Sanity Check

Consider a sequence consisting of random letters

What is the perplexity of this sentence according to a model that assign prob. 1/26 to each letter?

$$PP(w_1\, w_2 \ldots w_N) = \sqrt[N]{26^N}$$

$$= 26.$$

Moral: PP() measure makes sense for arbitrarily long sentences.

Low perplexity = Better Model

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	***962***	***170***	***109***

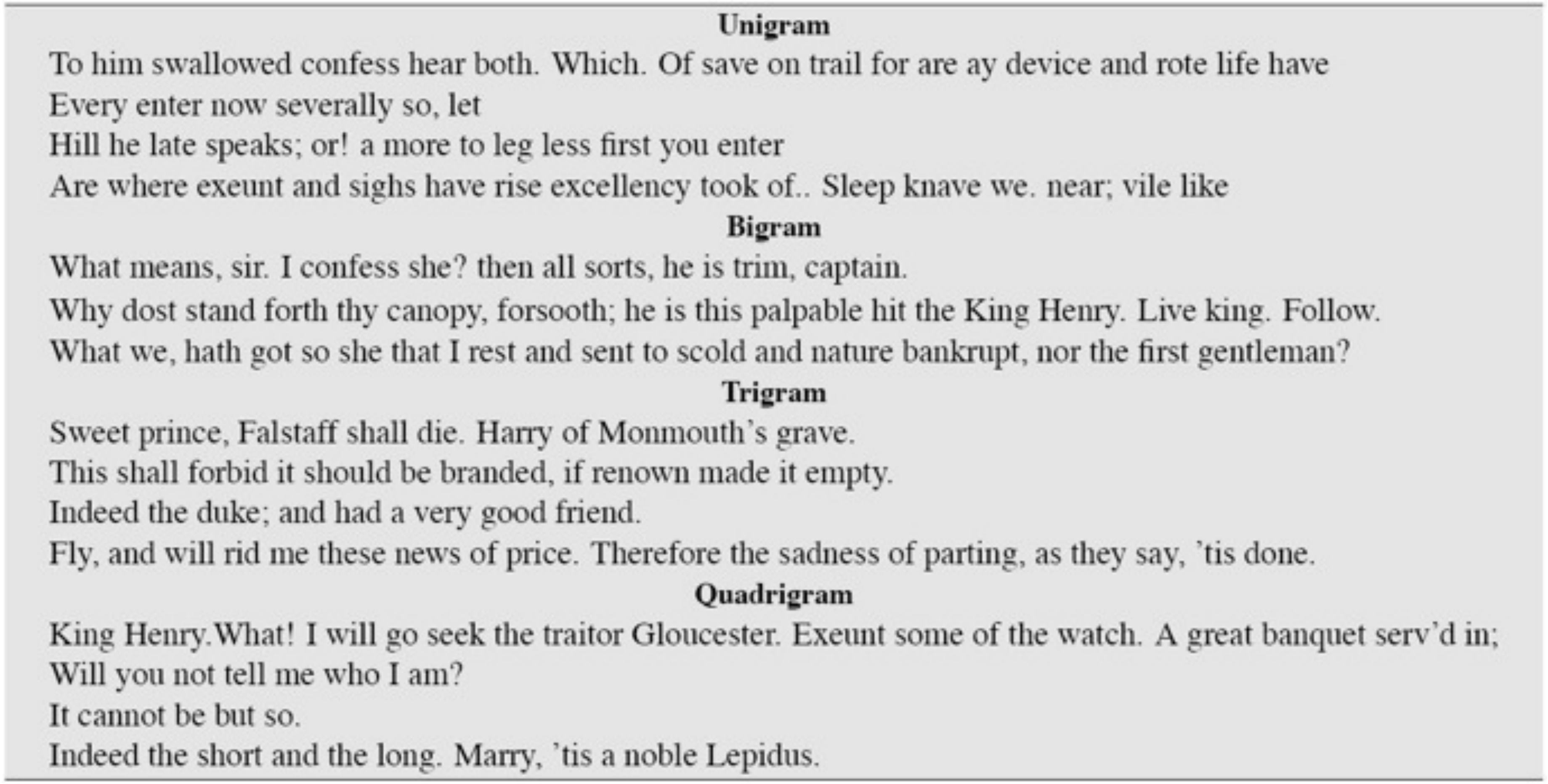
Language model gives a language generator

- Choose a random bigram

  (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

N-gram approximations to Shakespeare



Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

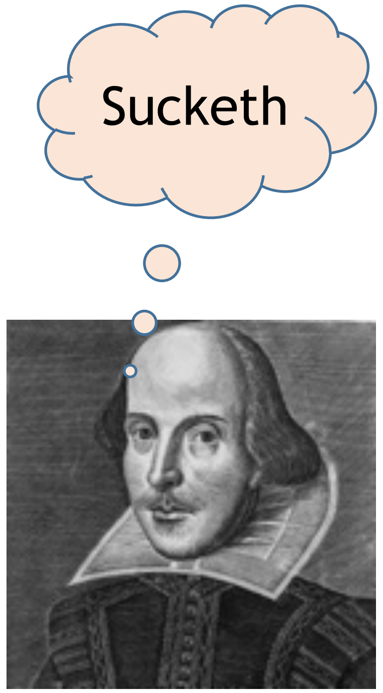
What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, 'tis a noble Lepidus.

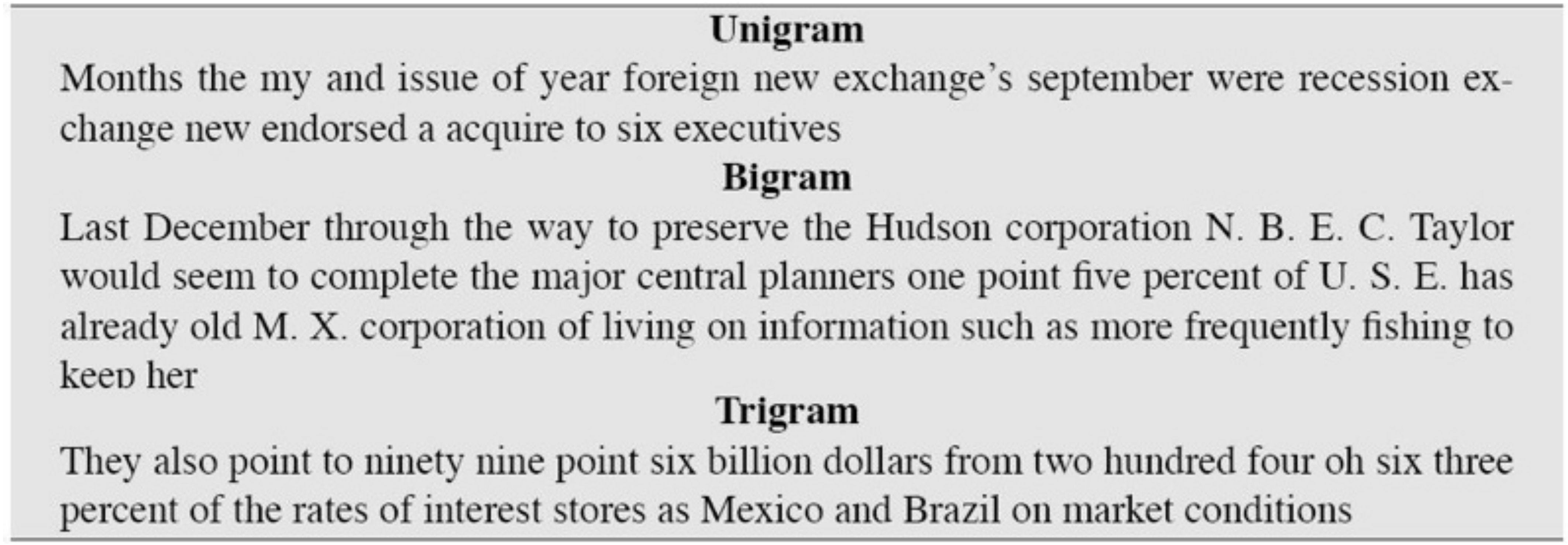


Why this even works (sort of)

- 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)
- Quadrigrams worse: What's coming out looks like Shakespeare because it ***is*** fragments of Shakespeare

Overfitting??? Will a model learnt using Shakespeare be any good for Dickens?

Wall Street Journal ≠ Shakespeare



Unigram

Months the my and issue of year foreign new exchange's september were recession ex-change new endorsed a acquire to six executives

Bigram

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

Trigram

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Suppose some bigram doesn't appear in training data,
but it appears in some sentence in **test data**.
What perplexity does the bigram model give to this sentence?

P(sentence) = 0.

➔ Perplexity = 1/0 = ∞

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

Example of a more general issue in finite sampling

You arrive in a new country with N people, and ask 5 randomly chosen people their
names: They are Joe, Shmoe, Doe, Roe, and Moe.

"Maybe all people in this country are called Joe/Shmoe/Doe/Roe/Moe"

"Maybe every person has a different name, and I got a random sample of 5 of them. "

Statisticians assign probabilities in such cases using Bayesian priors (won't do it to

"Smoothing trick": Reduce probability estimate of seen events; increase for unseen
events.

Add-one smoothing (aka Laplace smoothing)

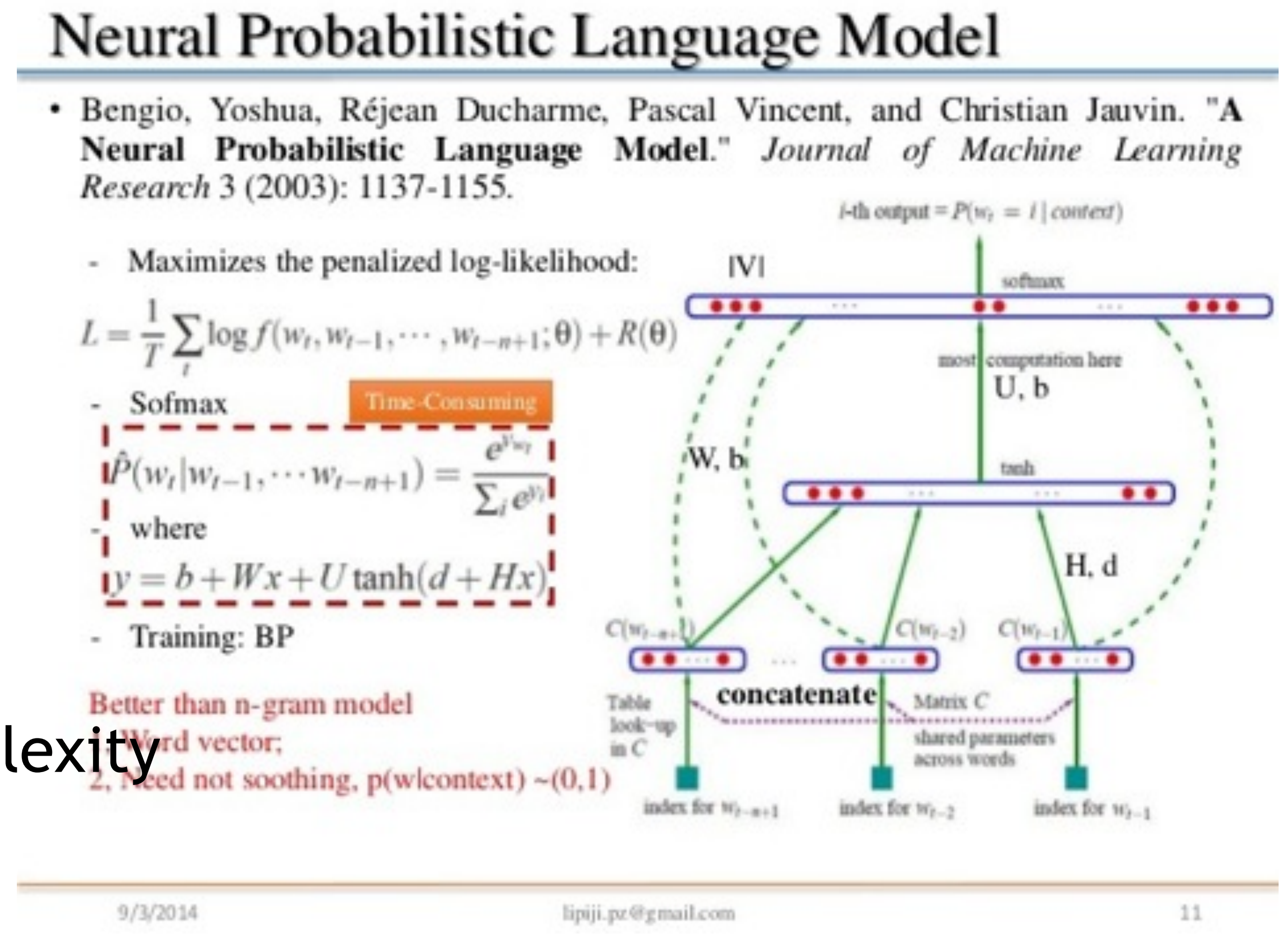
Many other smoothings invented : Good-Turing, Kneser-Ney, etc.

- Pretend we saw each word one more time than we
- "Add 1 to all bigram counts, and V to all unigram counts."
- Old estimate: $P_{MLE}(w_i \mid w_{i-1}) = \dfrac{c(w_{i-1}, w_i)}{c(w_{i-1})}$
- Add-1 estimate: $P_{Add-1}(w_i \mid w_{i-1}) = \dfrac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$

NB: For every word w, this doesn't affect $\sum_i P(w_i \mid w)$
(so we still have a proper distribution)

Advanced topic: Neural language models
(great progress in machine translation, question answering etc.)

Basic idea: Neural network represents language model but more compactly (fewer parameters).



Neural Probabilistic Language Model

- Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. "**A Neural Probabilistic Language Model**." *Journal of Machine Learning Research* 3 (2003): 1137-1155.
  - Maximizes the penalized log-likelihood:

$$L = \frac{1}{T}\sum_t \log f(w_t, w_{t-1}, \cdots, w_{t-n+1}; \theta) + R(\theta)$$

  - Sofmax

Time-Consuming

$$\hat{P}(w_t | w_{t-1}, \cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

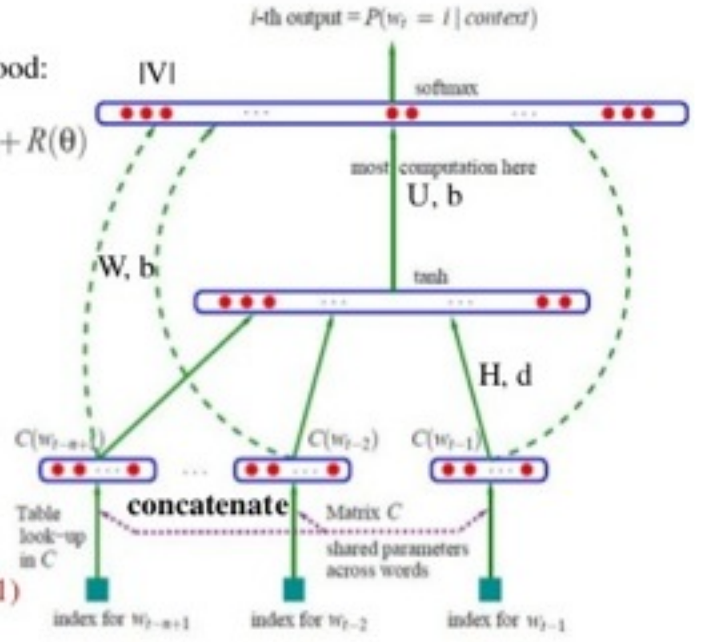
  - where

$$y = b + Wx + U \tanh(d + Hx)$$

  - Training: BP

Better than n-gram model
Word vector;
2, Need not soothing, p(w|context) ~(0,1)

9/3/2014 lipiji.pz@gmail.com 11



Training objective resembles perplexity
“Given last n words, predict the next with good probability.”

Next time: Attempts to capture semantics (meaning of words, sentences etc

(Vector-space models; also unsupervised learning)