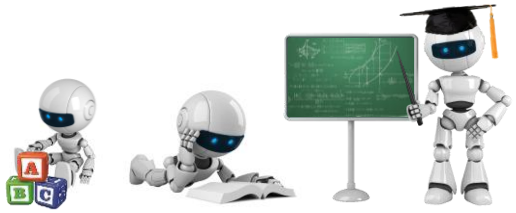
CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department

Slides are adopted from slides by Jurafsky and Martin



Logistical Comments

- HW3: Oct 11
- Project Status Report: Nov 3

- Dividing tasks

- Expected novelty:

- Need to be beyond assignments
- Projects will be graded relatively. The more creative project will need less empirical exploration

Probabilistic Language Models

- Language model: assigns a probability to a given sentence:
  - Spell Correction

    The office is about fifteen minutes form my house

    P(about fifteen minutes **from**) >> P(about fifteen minutes **form**)
  - Speech Recognition

    P(I saw a van) >> P(eyes awe of an)

    P(I listen to Pat and I see Mark) >> P(I listen to Pa Tennessee Mark)
  - Machine Translation:

    P(**high** winds tonight) > P(**large** winds tonight)

    P(**Put on** your shoes) > P(**Wear** your shoes)

Probabilistic Language Models

- Goal: compute the probability of a given 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 $P(W)$ or $P(w_n | w_1,w_2 \dots w_{n-1})$ is called a **language model**
- **language model** or **LM** is the standard term, but in a sense, we are modeling **the grammar** of the language

Probabilistic Language Models

- How can we compute a probability for a sentence

- Naïve idea: prepare a training dataset? (infeasible)

- We can use the sequential property of a sentence:

$$P(w_1,w_2,w_3,\ldots,w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1,w_2)\ldots P(w_n|w_1,\ldots,w_{n-1})$$

P("its water is so transparent") = P(its) × P(water|its) × P(is|its, water) × P(so|its, water, is) × P(transparent|its, water, is, so)

$$P(w_n|w_1,\ldots,w_{n-1}) = \frac{Count(w_1,\ldots,w_n)}{Count(w_1,\ldots,w_{n-1})}$$

Ex: P(Machine| "One of the AI branches is") = 2/7

Using Google and the Web

- Too many possible sentences!

Ex: (count("One of the AI branches is NLP" = 0))

- We'll never see enough data for estimating all possibilities

Language Models with Local Dependencies

- Markov simplification

$P(w_n|w_1,w_2 \ldots w_{n-1}) = P(w_n|w_{n-1})$

P(NLP| "One of the AI branches is") = P(NLP| "is")

=135,000 /23,440,000,000

- More generally for:

$$P(w_1,w_2\ldots,w_n) = \prod_{i=1}^{n} P(w_i|w_1,w_2\ldots w_{i-1}) = \prod_{i=1}^{n} P(w_i|w_k,w_{k+1\ldots}w_{i-1})$$

Unigram

- We consider full independence:

$$P(w_1,w_2 \ldots w_n) = \prod_{i=1}^{n} P(w_i)$$

- 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
```

Bigram

- We consider H=1:

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

- Some automatically generated sentences from a unigram model

```
texaco, rose, one, in, this, issue, is, pursuing, growth, in,
a, boiler, house, said, mr., gurria, mexico, 's, motion,
control, proposal, without, permission, from, five, hundred,
fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november
```

N-Grams

- We can extend to trigrams, 4-grams, …
- N-grams are not good generative model of language because language data has **long-distance dependencies**:

  "The computer which I had just put into the machine room on the fifth floor crashed."

- But we can often use N-gram models to perform downstream tasks, e.g., spellcheck, machine translation

Computing Bigram Probabilities

- Maximum likelihood estimation:

- Build a training dataset
- Estimate the probabilities using frequencies

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

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

Computing Bigram Probabilities

- Example:

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

Corpus

<s> I am Sam </s>

<s> Sam I am </s>

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

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

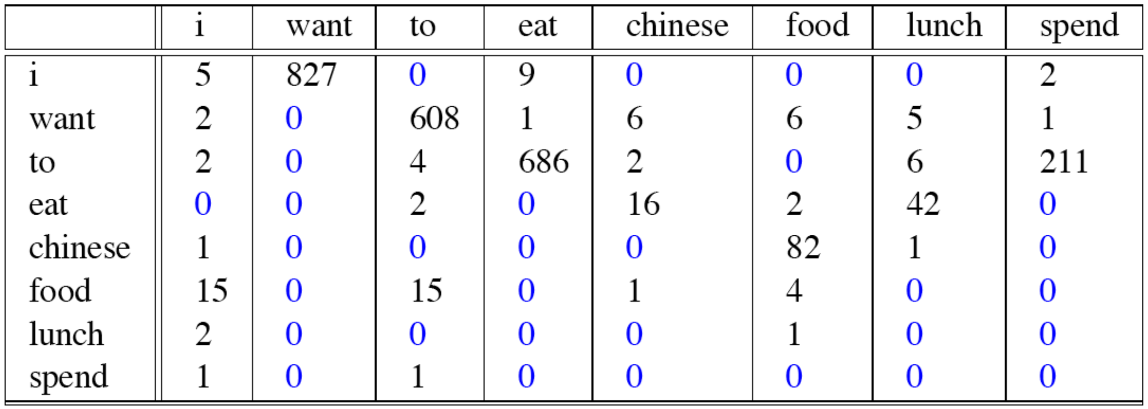
$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $\quad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $\quad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

Berkeley Restaurant Project

- Presented at ICLSP-94: spoken language turn into text data (9222 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

- 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

Berkeley Restaurant Project

- Bigram Probabilities

Normalize by unigram counts:

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

Resulting probabilities:

	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

Estimating probabilities of sentences:

P(<s> I want chinese food </s>) = P(I|<s>) × P(want|I) × P(chinese|want) × P(food|chinese) × P(</s>|food)= P(I|<s>) * 0.33 * 0.0065*0.52 * P(</s>|food)

P(<s> I chinese want food </s>) = P(I|<s>) × P(chinese|I) × P(want|chinese) × P(food|want) × P(</s>|food)= P(I|<s>) * 0.0063 * 0* 0.0065 * P(</s>|food)

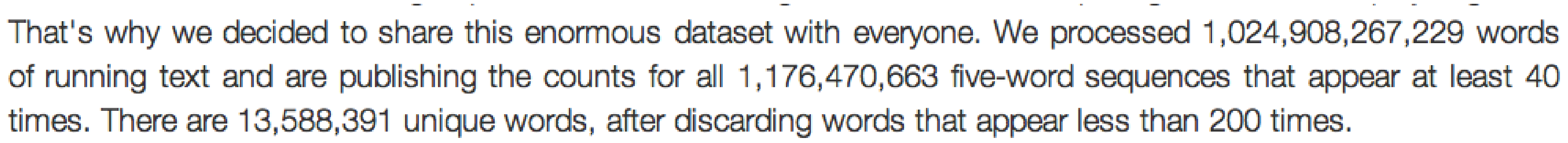
Accessible Language Models

- We do computations in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

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

- Language Modeling Toolkits

- SRILM: http://www.speech.sri.com/projects/srilm/
- KenLM:https://kheafield.com/code/kenlm/
- NLTK: https://www.nltk.org/api/nltk.lm.html
- Google: https://ai.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html



That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Language Model Evaluation

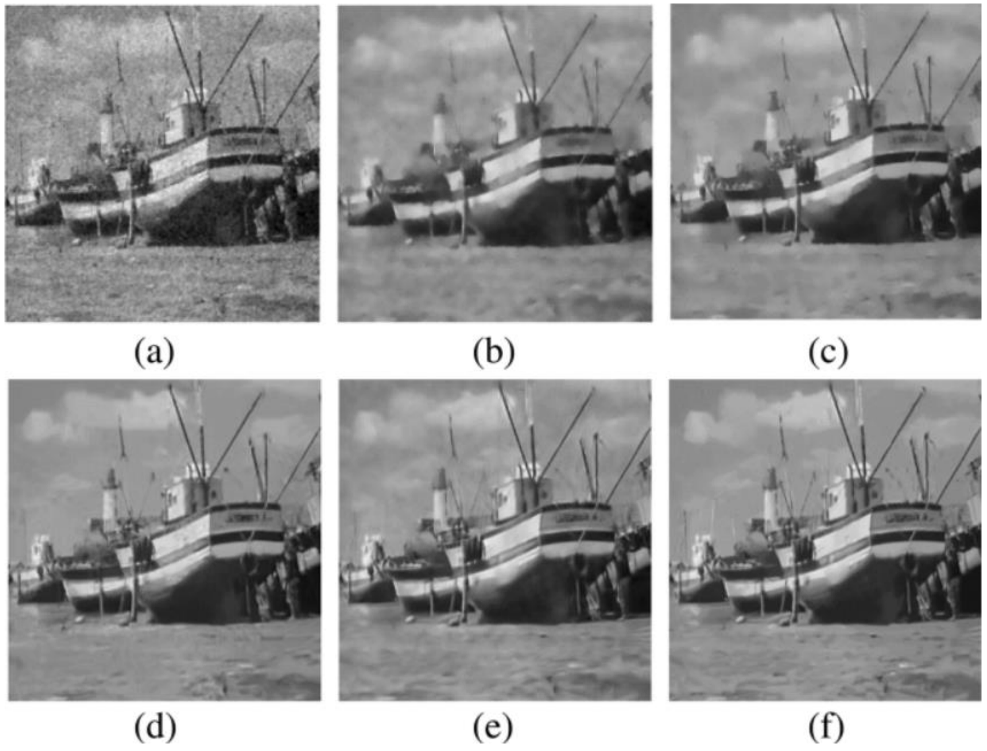
- What is a good language model?

- Assigns higher probability to “corrent” or “frequently observed” sentences Than “ungrammatical” or “rarely observed” sentences?

- We train parameters of our model on a training set and test the model’s performance on a testing set
- An **evaluation metric** is used to measure how well our model does on the test set.

Metrics

- Full Reference: Extrinsic



- Blind Metric: Intrinsic

Extrinsic Evaluation of N-Gram Models

- We can compare two models A and B by testing each model on a common task

- spelling corrector, speech recognizer, MT system
- Run the task, get an accuracy for A and for B

  How many misspelled words corrected properly?

  How many words translated correctly?

- Compare accuracy for A and B

- Extrinsic evaluation is time-consuming and depends on a task that may take a long time for completion

Intrinsic Evaluation of N-Gram Models

- **Perplexity**

- Only an approximation
  - the test data should look like the training data
  - generally only useful in pilot experiments (still helpful)

- Intuition: the best language model is the one that on an unseen test set

- Gives the highest P(sentence)
- Lower perplexity = better model

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

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

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

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

For bigrams

Intrinsic Evaluation of N-Gram Models

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

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

N-Gram Generalization

- 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 Generalization

- 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 Generalization

- 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*** Shakespeare

N-Gram Generalization

- The Wall Street Journal

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	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
3 gram	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

N-Gram Generalization

- N-grams only work well for word prediction if the test corpus looks like the training corpus
  - In real life, this often is not the case
  - We need to train robust models that generalize!
  - One kind of generalization: Zeros!
    - Things that don't ever occur in the training set
      - But occur in the test set

- Training set:
  … denied the allegations
  … denied the reports
  … denied the claims
  … denied the request

- Test set
  … denied the offer
  … denied the loan

P("offer" | denied the) = 0

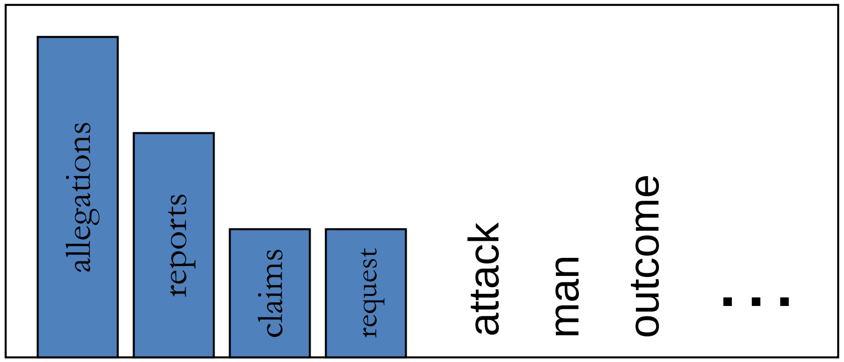
Smoothing

- Bigrams with zero probability
  - will assign 0 probability to the test set!
- Perplexity cannot be computed (can't divide by 0)!

- When we have sparse statistics:

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

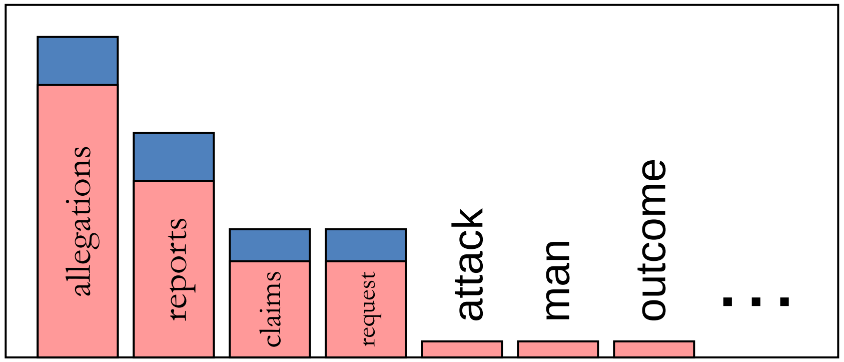
  7 total



- 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



Laplace Smoothing

- Pretend we saw each word one more time than we did
- Just add one to all the counts!

- MLE estimate:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Add-1 estimate:

$$P_{Add\text{-}1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

Berkeley Restaurant Corpus: Laplace 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

	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

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

- Reconstitutes counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

- Suitable for domains where the number of zeros isn't so huge
- Domains with high-level tasks, e.g., text classification.

	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

	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