

# Live Lecture – Week 3

## Agenda:

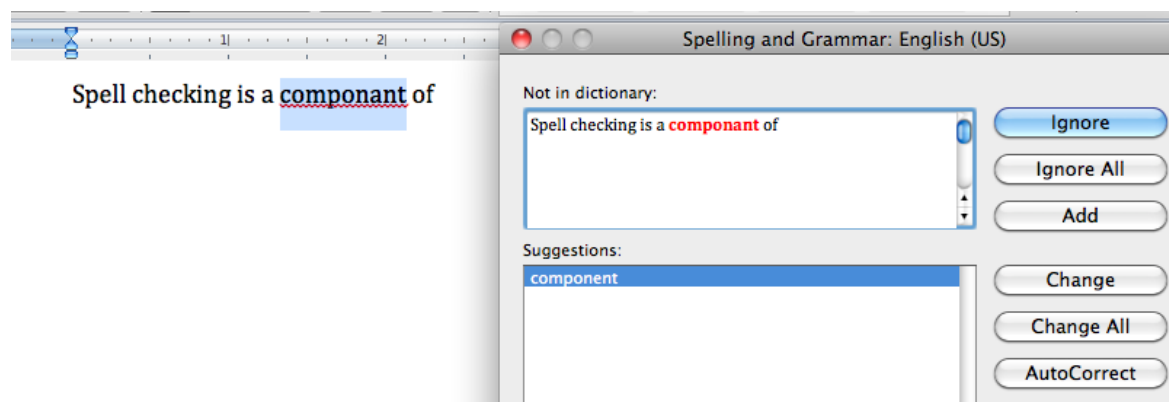
- Revising last week
- Preview for this week & next week
- Special topic: **N-gram** – to help understand the topics for this week, esp. spell correction.

## Note:

- Assignment is released today (25<sup>th</sup> Sept) and due on Tuesday 8<sup>th</sup> Oct, 5:00pm

# Applications for spelling correction

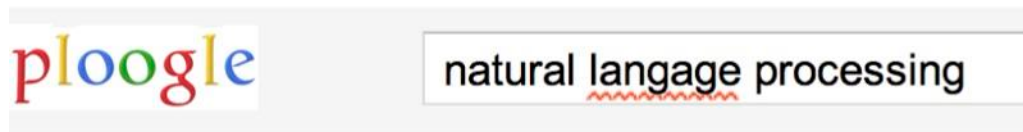
Word processing



Phones



Web search



Showing results for natural language processing  
 Search instead for natural language processing

# Language Modeling

## Introduction to N-grams

# Probabilistic Language Models

Goal: assign a probability to a sentence

- Machine Translation:
  - $P(\text{high winds tonite}) > P(\text{large winds tonite})$
- Spell Correction
  - The office is about fifteen **minuets** from my house
    - $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
- Speech Recognition
  - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$
- + Summarization, question-answering, etc., etc.!!

Why?

# 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**.

Better: **the grammar** But **language model** or **LM** is standard



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

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

$P(\text{"its water is so transparent"}) =$   
 $P(\text{its}) \times P(\text{water} \mid \text{its}) \times P(\text{is} \mid \text{its water})$   
 $\times P(\text{so} \mid \text{its water is}) \times P(\text{transparent} \mid \text{its water is so})$



# How to estimate these probabilities

Could we just count and divide?

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

No! Too many possible sentences!

We'll never see enough data for estimating these

# Markov Assumption



Andrei Markov

Simplifying assumption:

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

Or maybe

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

## 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

# Bigram model

- Condition on the previous word:

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

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-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

# Language Modeling

## Estimating N-gram Probabilities

# Estimating bigram probabilities

The Maximum Likelihood Estimate

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

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



## An example

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

<s> I am Sam </s>

<s> Sam I am </s>

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

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

$$P(\text{Sam} | \text{<s>}) = \frac{1}{3} = .33$$

$$P(\text{am} | \text{I}) = \frac{2}{3} = .67$$

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

$$P(\text{Sam} | \text{am}) = \frac{1}{2} = .5$$

$$P(\text{do} | \text{I}) = \frac{1}{3} = .33$$



## More examples: 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

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

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

$$\begin{aligned} P(<s> \text{ I want english food } </s>) = \\ & P(\text{I} | <s>) \\ & \times P(\text{want} | \text{I}) \\ & \times P(\text{english} | \text{want}) \\ & \times P(\text{food} | \text{english}) \\ & \times P(</s> | \text{food}) \\ & = .000031 \end{aligned}$$



# What kinds of knowledge?

$P(\text{english} | \text{want}) = .0011$

$P(\text{chinese} | \text{want}) = .0065$

$P(\text{to} | \text{want}) = .66$

$P(\text{eat} | \text{to}) = .28$

$P(\text{food} | \text{to}) = 0$

$P(\text{want} | \text{spend}) = 0$

$P(i | \langle s \rangle) = .25$

## Practical Issues

We do everything in log space

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

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

# Language Modeling

Smoothing: Add-one  
(Laplace) smoothing



## The intuition of smoothing (from Dan Klein)

When we have sparse statistics:

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

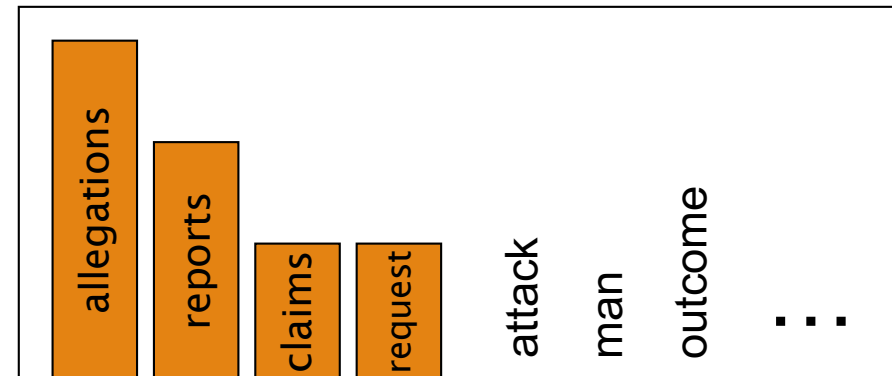
3 allegations

2 reports

1 claims

1 request

7 total



Steal probability mass to generalize better

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

2.5 allegations

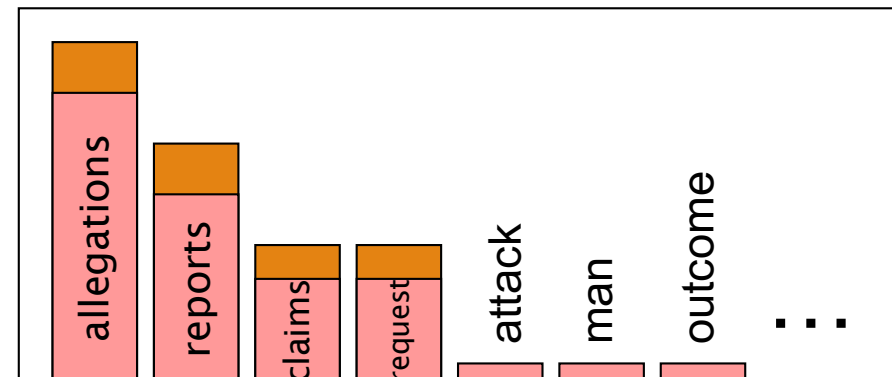
1.5 reports

0.5 claims

0.5 request

2 other

7 total



# Add-one estimation

Also called Laplace smoothing

Pretend we saw each word one more time than we did

Just add one to all the counts!

MLE estimate:

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

Add-1 estimate:

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

# Maximum Likelihood Estimates

The maximum likelihood estimate

- of some parameter of a model  $M$  from a training set  $T$
- maximizes the likelihood of the training set  $T$  given the model  $M$

Suppose the word “bagel” occurs 400 times in a corpus of a million words

What is the probability that a random word from some other text will be “bagel”?

MLE estimate is  $400/1,000,000 = .0004$

This may be a bad estimate for some other corpus

- But it is the **estimate** that makes it **most likely** that “bagel” will occur 400 times in a million word corpus.

## 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

# Laplace-smoothed bigrams

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

	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