Lecture 3
Words & Probabilities & (start of) LMs

CS 490A, Fall 2021 - Sept. 14

https://people.cs.umass.edu/~brenocon/cs490a_f21/

Laure Thompson and Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

including slides from SLP3

- **Python tutorial later today!!!**
  details on Piazza
- Finish up your HW0!
- Exercise today - please upload by Monday
- Next week, we'll release HW1: word probabilities & text classification

Words & Probabilities

- Today: from text,
  - 1. Detect a *single* word you want
  - 2. Get a string of words
  - 3. Analyze word probabilities
  - 4. Model *text* probability: language models

Example

- Find me all instances of the word “the” in a text.

`the`

Misses capitalized examples

`[tT]he`

Incorrectly returns `other` or `theology`

`[^a-zA-Z][tT]he[^a-zA-Z]`

Errors

- The process we just went through was based on fixing two kinds of errors
  - Matching strings that we should not have matched (there, then, other)
    - False positives (Type I)
  - Not matching things that we should have matched (The)
    - False negatives (Type II)

Errors cont.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
  - Increasing accuracy or precision (minimizing false positives)
  - Increasing coverage or recall (minimizing false negatives).

Text normalization + tok.

- Every NLP task needs text normalization

  - 1. Segment/tokenize words in running text
    - (OK in English, but not as simple preproc in many langs)

  - 2. Normalizing word formats

  - 3. Sentence segmentation
    and/or
    paragraphs/sections/chapters/etc.

- Demo: simple tokenization in python

Text preprocessing & data cleaning

```
DISCHARGE CONDITION:  The patient was able to oxygenate
on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.

DISCHARGE STATUS:  The patient will be discharged to
[**Hospital1 **]
for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
```

Text data
(MIMIC III EHR)

All-caps headers delineate sections: should be parsed out as structure

Unstructured, linguistic data
Has *semantic* structure: describes properties and relationships among entities

```
DISCHARGE CONDITION:  The patient was able to oxygenate
on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.

DISCHARGE STATUS:  The patient will be discharged to
[**Hospital1 **]
for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
```

Text data
(MIMIC III EHR)

Semi-structured, regular ordering

MEDICINE_NAME NUMBER UNITS MODIFIERS

All-caps headers delineate sections: should be parsed out as structure

Unstructured, linguistic data
Has *semantic* structure: describes properties and relationships among entities

```
DISCHARGE CONDITION:  The patient was able to oxygenate
on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.

DISCHARGE STATUS:  The patient will be discharged to
[**Hospital1 **]
for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
```

Semi-structured, regular ordering

MEDICINE_NAME NUMBER UNITS MODIFIERS

Text data (MIMIC III EHR)

Easy to structure: write hard-coded, custom string processor

Harder: develop more complex processor

Hardest: Natural language
Many possible end goals

Summary

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalizations

[Slide: SLP3]

Preprocessing: Text cleaning

```
DISCHARGE CONDITION:  The patient was able to oxygenate
on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.

DISCHARGE STATUS:  The patient will be discharged to
[**Hospital1 **]
for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
```

```
The patient was able to oxygenate on
room air at 93% at the time of discharge.  She was profoundly
weak, but was no longer tachycardic and had a normal blood
pressure.  Her respirations were much improved albeit with
transmitted upper airway sounds.
```

- Remove unwanted structure to just the sentences/ paragraphs you want to analyze
- Get text out of weird formats like HTML, PDFs, or idiosyncratic formatting (e.g. in these EHRs)

This step is usually specific to your dataset

Preprocessing: Tokenization

```
The patient was able to oxygenate on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.
```

```
['The', 'patient', 'was', 'able', 'to', 'oxygenate',
'on', 'room', 'air', 'at', '93', '%', 'at', 'the',
'time', 'of', 'discharge', '.', 'She', 'was',
'profoundly', 'weak', ',', 'but', 'was', 'no',
'longer', 'tachycardic', 'and', 'had', 'a',
'normal', 'blood', 'pressure', '.', 'Her',
'respirations', 'were', 'much', 'improved',
'albeit', 'with', 'transmitted', 'upper', 'airway',
'sounds', '.']
```

- Words are (usually) the basic units of analysis in NLP.
- In English, words are delineated as tokens via space and punctuation conventions, recognizable via moderately simple rules
- Tokenization: from text string to sequence of word strings
- Sentence splitting: harder but sometimes done too

There are good off-the-shelf tokenizers (NLTK, SpaCy, CoreNLP, Twokenizer)

Preprocessing: Normalization

- Often:
  - Lowercase words (“She” -> “she”)
- Sometimes:
  - Remove numbers (“93” -> “NUMBER_NN”)
  - Correct misspellings / alternate spellings (“color” -> “colour”)
- Problem specific:
  - Resolve synonyms / aliases (if you know them already)
  - Remove “stopwords”
    - Punctuation and grammatical function words (“if”, “the”, “by”), and
    - Very common words in your domain that don’t add much meaning

Issues in Tokenization

- `Finland's capital → Finland Finlands Finland's ?`
- `what're, I'm, isn't → What are, I am, is not`
- `Hewlett-Packard → Hewlett Packard ?`
- `state-of-the-art → state of the art ?`
- `Lowercase → lower-case lowercase lower case ?`
- `San Francisco → one token or two?`
- `m.p.h., PhD. → ??`

[Slide: SLP3]

Tokenization: language issues

- French
  - ***L'ensemble*** → one token or two?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Want ***l'ensemble*** to match with ***un ensemble***

- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German information retrieval needs **compound splitter**

[Slide: SLP3]

Demo: word counts!

How many words?

N = number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

Church and Gale (1990): $|V| > O(N^{½})$

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

"word" is ambiguous.
"word token" vs "word type"
is crucial terminology in this course!!!

Zipf's Law

- When word types are ranked by frequency, then frequency (f) * rank (r) is roughly equal to some constant (k)

$$f \times r = k$$

Implications for NLP?

Lemmatization

- Reduce inflections or variant forms to base form
  - *am, are, is → be*
  - *car, cars, car's, cars' → car*
- *the boy's cars are different colors → the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation
  - Spanish quiero (‘I want’), quieres (‘you want’) same lemma as querer ‘want’

Morphology

- **Morphemes**:
  - The small meaningful units that make up words
  - **Stems**: The core meaning-bearing units
  - **Affixes**: Bits and pieces that adhere to stems
    - Often with grammatical functions

language modeling: n-gram models

CS 490A, Fall 2021

https://people.cs.umass.edu/~brenocon/cs490a_f21/

Laure Thompson and Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

including slides from Mohit Iyyer, Dan Jurafsky, Richard Socher

goal: assign probability to a piece of text

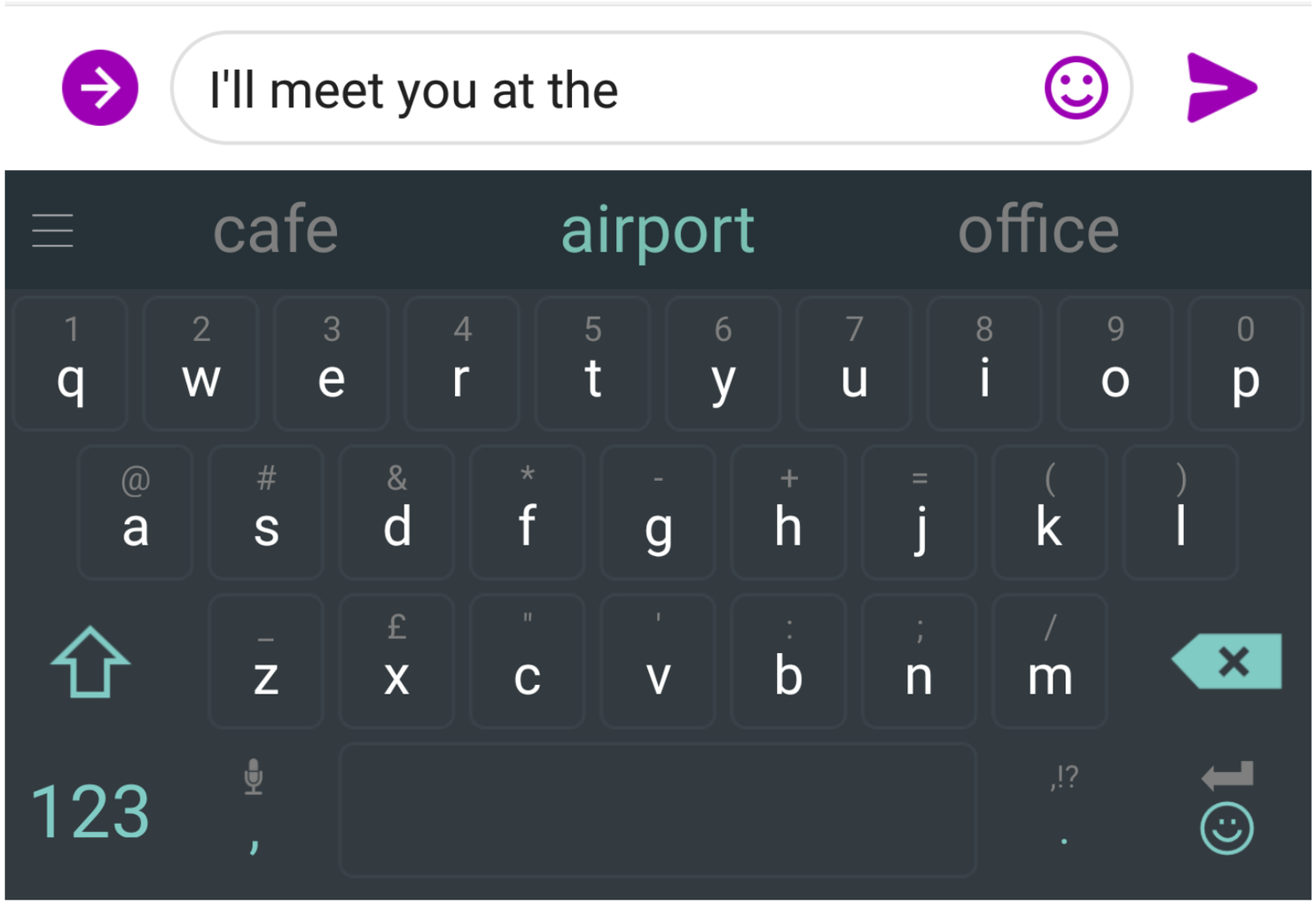
- why would we ever want to do this?

- translation:
  - P(i flew to the movies) <<<<< P(i went to the movies)

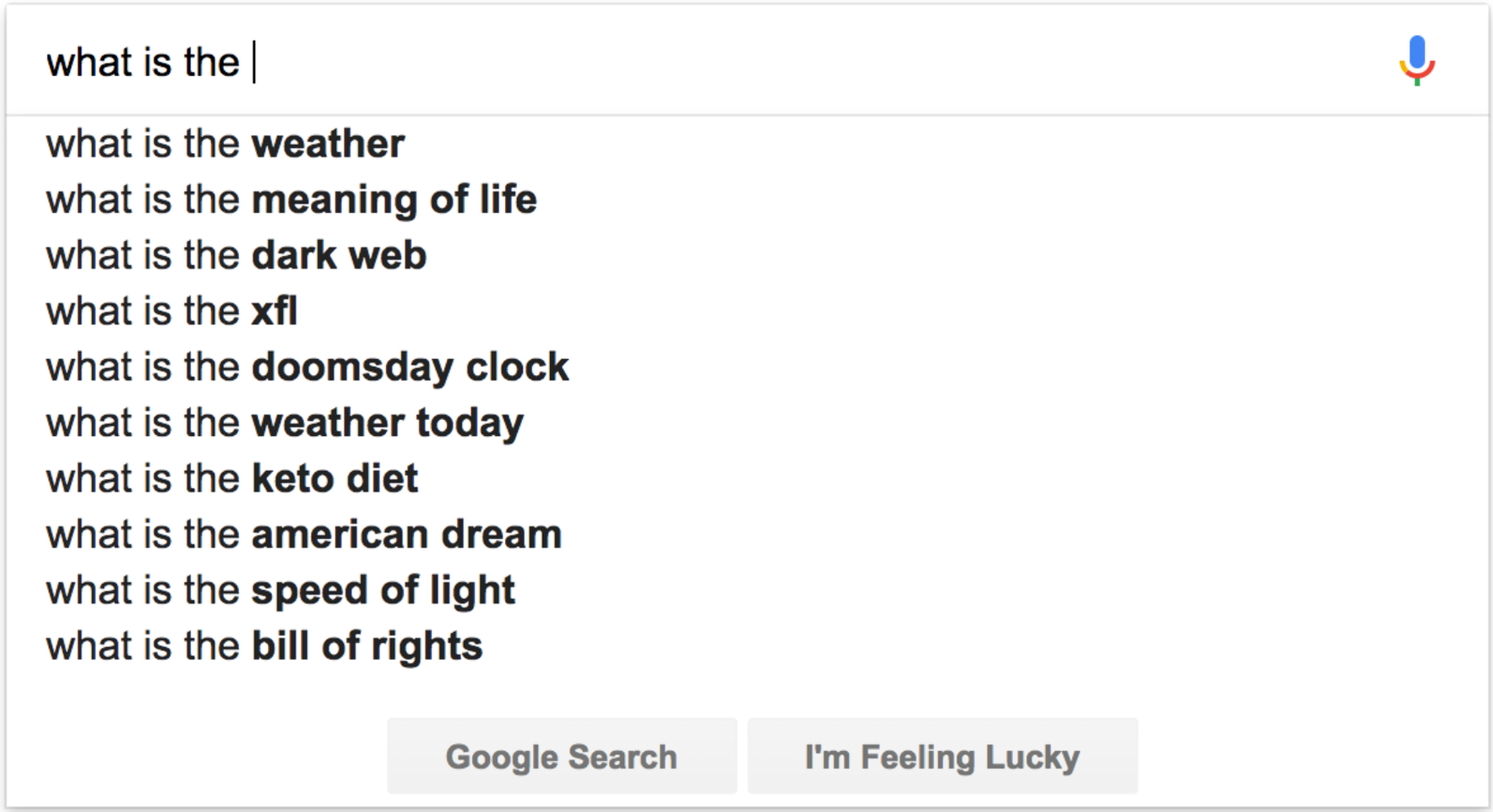
- speech recognition:
  - P(i saw a van) >>>>> P(eyes awe of an)

- text classification (next week):
  - P(i am so mad!! | *[author is happy]*) <
    P(i am so mad!! | *[author is not happy]*)

You use Language Models every day!



You use Language Models every day!



Probabilistic Language Modeling

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

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

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

we have already seen one way to do this… where?

How to compute P(W)

- How to compute this joint probability:
  - P(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) 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

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

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

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

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)

How to estimate these probabilities

- Could we just count and divide?

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

How to estimate these probabilities

- Could we just count and divide?

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

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

- stopped here 9/9

Markov Assumption

Andrei Markov (1856~1922)

- **Simplifying assumption:**

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

- Or maybe

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

Markov Assumption

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

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

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

Simplest case: Unigram model

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

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
```

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

we're doing longer-distance language modeling near the end of this course

Estimating bigram probabilities

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

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

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

c – count

An example

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

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

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

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $\quad$ $P(\texttt{Sam} \mid \texttt{am}) = ???$

An example

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

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

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

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

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

- 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

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

- 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> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

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

What kinds of knowledge?

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66

about the world

grammar – infinitive verb

- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

???

grammar

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!
- And violates the honor code

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

Zeros

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

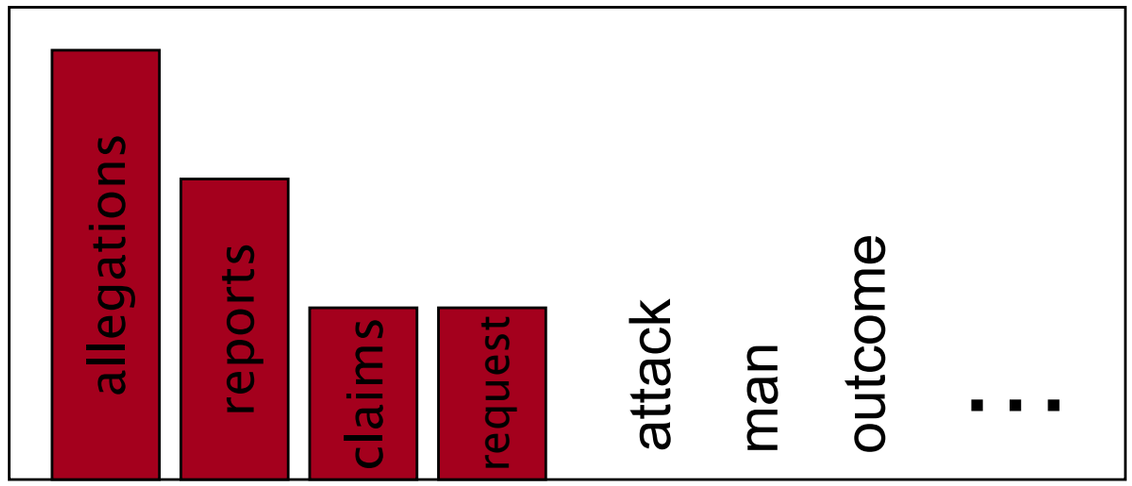
P(“offer” | denied the) = 0

- Test set
  … denied the offer
  … denied the loan

The intuition of smoothing (from Dan Klein)

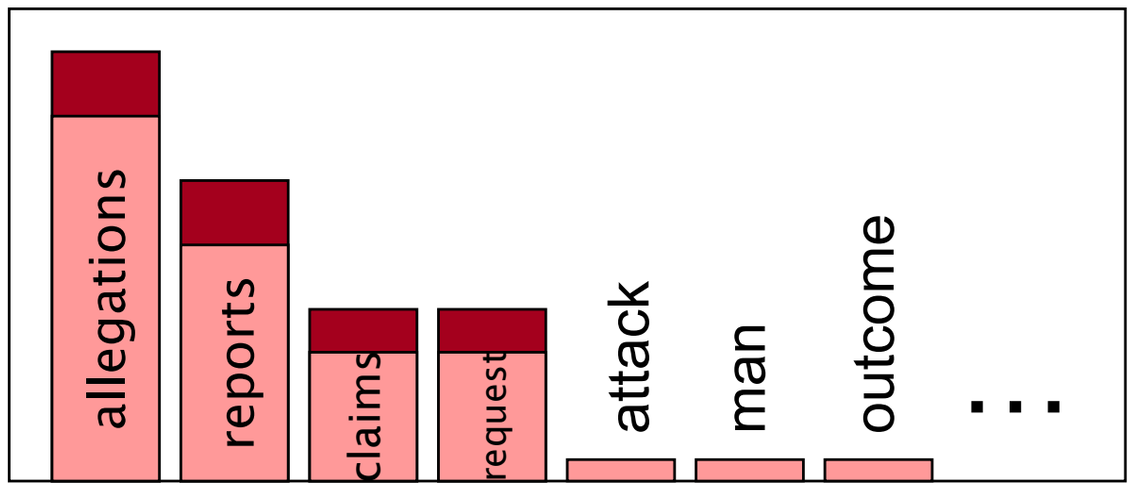
- When we have sparse statistics:

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



Add-one estimation (again!)

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

- MLE estimate:

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

- Add-1 estimate:

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

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

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

Reconstituted counts

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

	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

Compare with 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

	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

Add-1 estimation is a blunt instrument

- So add-1 isn't used for N-grams:
  - We'll see better methods
- But add-1 is used to smooth other NLP models
  - For text classification
  - In domains where the number of zeros isn't so huge.

Backoff and Interpolation

- Sometimes it helps to use **less** context
  - Condition on less context for contexts you haven't learned much about
- **Backoff:**
  - use trigram if you have good evidence,
  - otherwise bigram, otherwise unigram
- **Interpolation:**
  - mix unigram, bigram, trigram

- Interpolation works better

Linear Interpolation

- Simple interpolation

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

$$\sum_i \lambda_i = 1$$

- Lambdas conditional on context:

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1(w_{n-2}^{n-1}) P(w_n|w_{n-2}w_{n-1}) + \lambda_2(w_{n-2}^{n-1}) P(w_n|w_{n-1}) + \lambda_3(w_{n-2}^{n-1}) P(w_n)$$

Absolute discounting: just subtract a little from each count

- Suppose we wanted to subtract a little from a count of 4 to save probability mass for the zeros
- How much to subtract ?

- Church and Gale (1991)'s clever idea
- Divide up 22 million words of AP Newswire
  - Training and held-out set
  - for each bigram in the training set
  - see the actual count in the held-out set!

Bigram count in training	Bigram count in heldout set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

Absolute discounting: just subtract a little from each count

- Suppose we wanted to subtract a little from a count of 4 to save probability mass for the zeros
- How much to subtract ?

- Church and Gale (1991)'s clever idea
- Divide up 22 million words of AP Newswire
  - Training and held-out set
  - for each bigram in the training set
  - see the actual count in the held-out set!

Bigram count in training	Bigram count in heldout set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

why do you think the training and heldout counts differ?

Absolute Discounting Interpolation

- Save ourselves some time and just subtract 0.75 (or some d)!

discounted bigram

Interpolation weight

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w)$$

unigram

- (Maybe keeping a couple extra values of d for counts 1 and 2)
- But should we really just use the regular unigram P(w)?