CS 4705

N-Grams and Corpus Linguistics

Questions from last time?

- The textbook is now available in Book Culture (112th St)
- Everyone in the class (both hybrid and in-person) can now find the videos on Canvas. Look for “Video Library” on left panel. Videos are available 24-48 hours after the class.
- Apparently the courseworks link is under Summer 2017
- The textbook, Jurafsky and Martin, is NOW available at Book Culture.

This class: Language Modeling

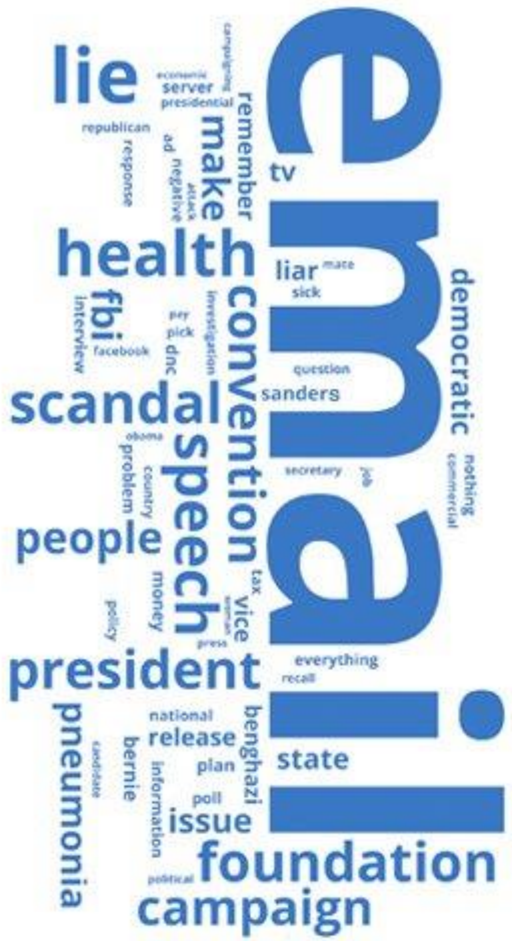
- Why is language modeling useful?
- Computing language models
- An example
- Evaluating language models
- Smoothing
- HW0

- “But it must be recognized that the notion of “probability of a sentence” is an entirely useless one, under any known interpretation of this term.” Noam Chomsky (1969)

- “Anytime a linguist leaves the group the recognition rate goes up.” Fred Jelinek (1988)

What Americans Have Heard or Read About Hillary Clinton

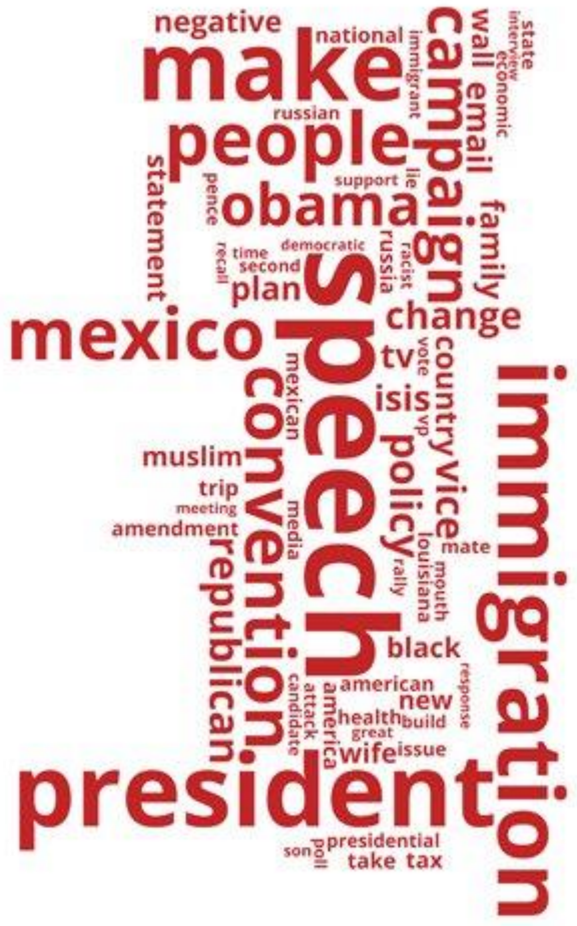
What specifically do you recall reading, hearing or seeing about Hillary Clinton in the last day or two?



GALLUP DAILY TRACKING
JULY 17-SEPT 18, 2016

What Americans Have Heard or Read About Donald Trump

What specifically do you recall reading, hearing or seeing about Donald Trump in the last day or two?



GALLUP DAILY TRACKING
JULY 17-SEPT 18, 2016

Next Word Prediction

- *Stocks plunged this ….*
- *Let’s meet in Times ….*
- *I took the subway to ….*

Response

Seinfeld

- I was in the city
- I was on the island
- I got on the train
- I got in the taxi

Next Word Prediction

- From a NY Times story...
  - Stocks plunged this ….
  - Stocks plunged this morning, despite a cut in interest rates
  - Stocks plunged this morning, despite a cut in interest rates by the Federal Reserve, as Wall ...
  - Stocks plunged this morning, despite a cut in interest rates by the Federal Reserve, as Wall Street began

Human Word Prediction

- Clearly, at least some of us have the ability to predict future words in an utterance.
- How?
  - Domain knowledge
  - Syntactic knowledge
  - Lexical knowledge

Slide from Dragomir Radev

Claim

- A useful part of the knowledge needed to allow Word Prediction can be captured using simple statistical techniques
- We'll rely on the notion of the probability of a sequence (of letters, words,...)

- Formula:
  - $P(w_n | w_1\ w_2\ w_3 \ldots W_{n-1})$

Slide from Dragomir Radev

Applications

- Speech recognition
  - P("*recognize speech*" > P("*wreck a nice beach*")
- Text generation
  - P("*three houses*") > P("*three house*")
- Spelling correction
  - P("*my cat eats fish*") > P("*my xat eats fish*")
- Machine Translation
  - P("*the blue house*") > P("*the house blue*")
- Other uses
  - OCR
  - Summarization
  - Document classification

Slide from Dragomir Radev

N-Gram Models of Language

- Markov assumption: Use the previous N-1 words in a sequence to predict the next word
- Language Model (LM)
  - unigrams, bigrams, trigrams,…
- How do we train these models?
  - Very large corpora

Slide from Dragomir Radev

Corpora

- Corpora are online collections of text and speech
  - Brown Corpus
  - Wall Street Journal
  - AP newswire
  - Hansards
  - DARPA/NIST text/speech corpora (Call Home, ATIS, switchboard, Broadcast News, TDT, Communicator)
  - TRAINS, Radio News

Slide from Dragomir Radev

Google 1-T Corpus

1 trillion word tokens

- Number of tokens – 1,024,908,267,229
- Number of sentences – 95,119,665,584
- Number of unigrams – 13,588,391
- Number of bigrams – 314,843,401
- Number of trigrams – 977,069,902
- Number of fourgrams – 1,313,818,354
- Number of fivegrams – 1,176,470,663

Slide from Dragomir Radev

Google N-Gram Release

- `serve as the incoming 92`
- `serve as the incubator 99`
- `serve as the independent 794`
- `serve as the index 223`
- `serve as the indication 72`
- `serve as the indicator 120`
- `serve as the indicators 45`
- `serve as the indispensable 111`
- `serve as the indispensible 40`
- `serve as the individual 234`

Slide from Dragomir Radev

Counting Words in Corpora

- What is a word?
  - e.g., are cat and cats the same word?
  - September and Sept?
  - zero and oh?
  - Is _ a word? * ? ‘(‘ ?
  - How many words are there in don’t ? Gonna ?
  - In Japanese and Chinese text -- how do we identify a word?

Slide from Dragomir Radev

Terminology

- Sentence: unit of written language
- Utterance: unit of spoken language
- Word Form: the inflected form as it actually appears in the corpus
- Lemma: an abstract form, shared by word forms having the same stem, part of speech, and word sense – stands for the class of words with stem
- Types: number of distinct words in a corpus (vocabulary size)
- Tokens: total number of words

Simple N-Grams

- Assume a language has T word types in its lexicon, how likely is word x to follow word y?
  - Simplest model of word probability: 1/T
  - Alternative 1: estimate likelihood of x occurring in new text based on its general frequency of occurrence estimated from a corpus (unigram probability)

    popcorn is more likely to occur than *unicorn*
  - Alternative 2: condition the likelihood of x occurring in the context of previous words (bigrams, trigrams,…)

    mythical unicorn is more likely than *mythical popcorn*

Computing the Probability of a Word Sequence

- Compute the product of component conditional probabilities?
  - P(the mythical unicorn) = P(the) P(mythical|the) * P(unicorn|the mythical)
- The longer the sequence, the less likely we are to find it in a training corpus

  P(Most biologists and folklore specialists believe that in fact the mythical unicorn horns derived from the narwhal)
- Solution: approximate using n-grams

Bigram Model

- Approximate $P(w_n|w_1^{n-1})$ by $P(w_n|w_{n-1})$
  - P(unicorn|the mythical) by P(unicorn|mythical)
- Markov assumption: the probability of a word depends only on the probability of a limited history
- Generalization: the probability of a word depends only on the probability of the ***n*** previous words
  - trigrams, 4-grams, …
  - the higher n is, the more data needed to train. Thus backoff models…

Using N-Grams

- For N-gram models
  - $P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$
  - $P(w_{n-1},w_n) = P(w_n \mid w_{n-1})\, P(w_{n-1})$
  - By the Chain Rule we can decompose a joint probability, e.g. $P(w_1,w_2,w_3)$

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

For bigrams then, the probability of a sequence is just the product of the conditional probabilities of its bigrams

P(the,mythical,unicorn) = P(unicorn|mythical) P(mythical|the) P(the|<start>)

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

Training and Testing

- N-Gram probabilities come from a training corpus
  - overly narrow corpus: probabilities don't generalize
  - overly general corpus: probabilities don't reflect task or domain
- A separate test corpus is used to evaluate the model, typically using standard metrics
  - held out test set; development (dev) test set
  - cross validation
  - results tested for statistical significance – how do they differ from a baseline? Other results?

A Simple Example

- P(I want to eat Chinese food) = P(I | <start>) P(want | I) P(to | want) P(eat | to) P(Chinese | eat) P(food | Chinese) P(<end>|food)

A Bigram Grammar Fragment from BERP

Eat on	.16	Eat Thai	.03
Eat some	.06	Eat breakfast	.03
Eat lunch	.06	Eat in	.02
Eat dinner	.05	Eat Chinese	.02
Eat at	.04	Eat Mexican	.02
Eat a	.04	Eat tomorrow	.01
Eat Indian	.04	Eat dessert	.007
Eat today	.03	Eat British	.001

<start> I	.25	Want some	.04
<start> I’d	.06	Want Thai	.01
<start> Tell	.04	To eat	.26
<start> I’m	.02	To have	.14
I want	.32	To spend	.09
I would	.29	To be	.02
I don’t	.08	British food	.60
I have	.04	British restaurant	.15
Want to	.65	British cuisine	.01
Want a	.05	British lunch	.01

- P(I want to eat British food) = P(I|<start>) P(want|I) P(to|want) P(eat|to) P(British|eat) P(food|British) = .25*.32*.65*.26*.001*.60 = .000080
  - Suppose P(<end>|food) = .2?
  - vs. *I want to eat Chinese food* = .00014 * ? *What is the formula for this sentence? You can answer in numbers only.*

Response

- P(I want to eat British food) = P(I|<start>) P(want|I) P(to|want) P(eat|to) P(British|eat) P(food|British) = .25*.32*.65*.26*.001*.60 = .000080
  - Suppose P(<end>|food) = .2?
  - vs. *I want to eat Chinese food* = .00014 * ? *What is the formula for this sentence? You can answer in numbers only.*
- Probabilities roughly capture ``syntactic'' facts, ``world knowledge''
  - eat is often followed by an NP
  - British food is not too popular
- N-gram models can be trained by counting and normalization

We can compute bigram probabilities using corpus counts
BERP Bigram Counts

	I	Want	To	Eat	Chinese	Food	lunch
I	8	1087	0	13	0	0	0
Want	3	0	786	0	6	8	6
To	3	0	10	860	3	0	12
Eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
Food	19	0	17	0	0	0	0
Lunch	4	0	0	0	0	1	0

BERP Bigram Probabilities

- Maximum Likelihood Estimation (MLE): relative frequency of e.g. $\frac{freq(w_1, w_2)}{freq(w_1)}$
- Normalization: divide each row's counts by appropriate unigram counts for w_{n-1}

I	Want	To	Eat	Chinese	Food	Lunch
3437	1215	3256	938	213	1506	459

- Computing the bigram probability of I I
  - P(I|I) = C(I,I)/C(all I)
  - p (I|I) = 8 / 3437 = .0023

What is the probability of "want to"?

	I	Want	To	Eat	Chinese	Food	lunch
I	8	1087	0	13	0	0	0
Want	3	0	786	0	6	8	6
To	3	0	10	860	3	0	12
Eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
Food	19	0	17	0	0	0	0
Lunch	4	0	0	0	0	1	0

I	Want	To	Eat	Chinese	Food	Lunch
3437	1215	3256	938	213	1506	459

Response

What do we learn about the language?

- *What's being captured with ...*
  - *P(to | want) = .65*
  - *P(eat | to) = .26*
  - *P(food | Chinese) = .56*
  - *P(lunch | eat) = .055*
- *What about...*
  - *P(I | I) = .0023*
  - *P(I | want) = .0025*
  - *P(I | food) = .013*

Response

- P(I | I) = .0023 I I I I want
- P(I | want) = .0025 I want I want
- P(I | food) = .013 the kind of food I want is ...

Approximating Shakespeare

- As we increase the value of N, the accuracy of an n-gram model increases, since choice of next word becomes increasingly constrained
- Generating sentences with random unigrams...
    - *Every enter now severally so, let*
    - *Hill he late speaks; or! a more to leg less first you enter*
- With bigrams...
    - *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.*

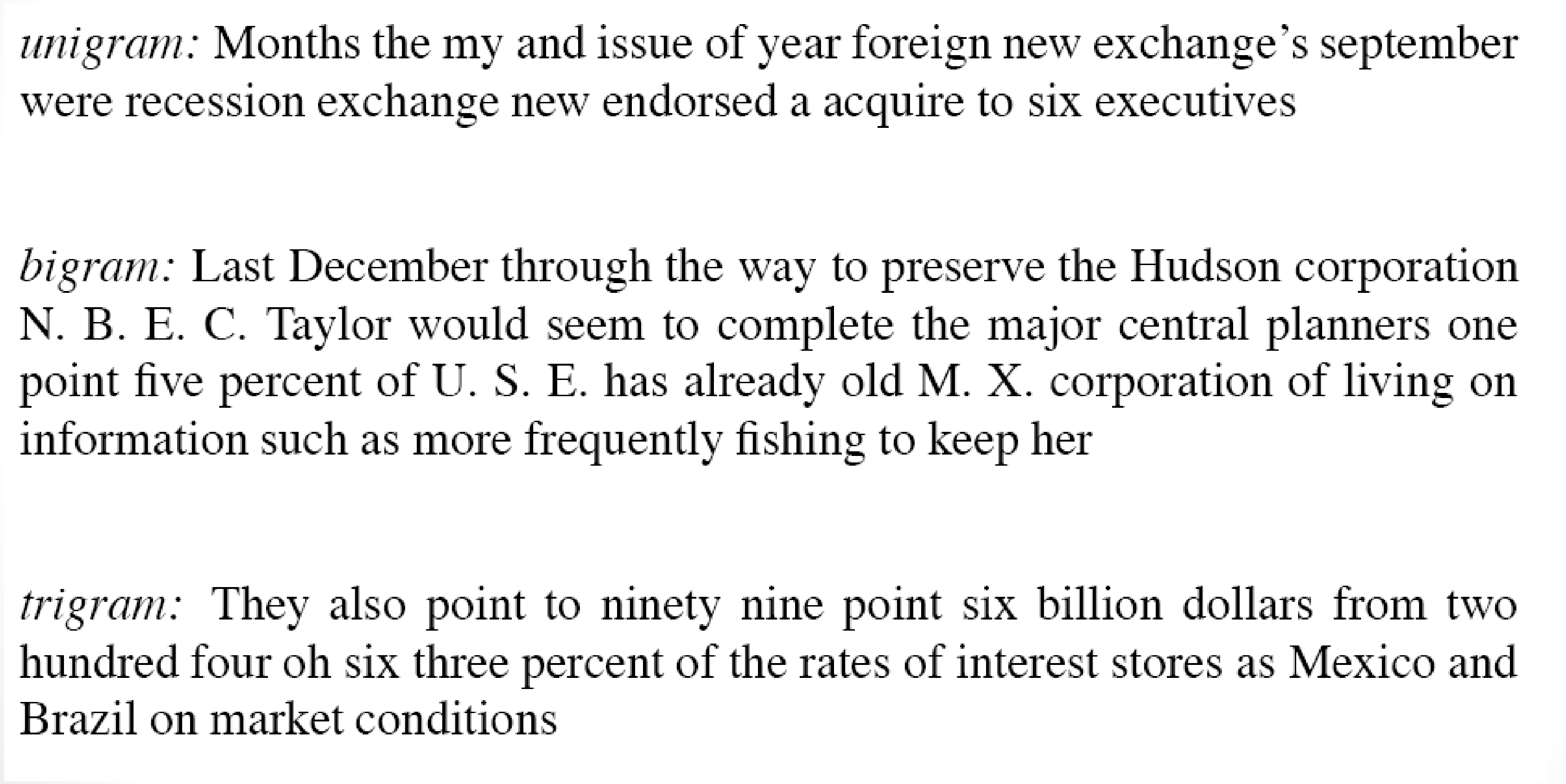
- Trigrams
  - *Sweet prince, Falstaff shall die.*
  - *This shall forbid it should be branded, if renown made it empty.*
- Quadrigrams
  - *What! I will go seek the traitor Gloucester.*
  - *Will you not tell me who I am?*

- There are 884,647 tokens, with 29,066 word form types, in an approximately one million word Shakespeare corpus
- Shakespeare produced 300,000 bigram types out of 844 million possible bigrams: so, 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams: What's coming out looks like Shakespeare because it ***is*** Shakespeare

N-Gram Training Sensitivity

- If we repeated the Shakespeare experiment but trained our n-grams on a Wall Street Journal corpus, what would we get?
- This has major implications for corpus selection or design

The wall street journal is *not* shakespeare



unigram: Months the my and issue of year foreign new exchange's september were recession exchange 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

Some Important Concepts

- Smoothing and Backoff : how do you handle unseen n-grams?
- Perplexity and entropy: how do you estimate how well your language model fits a corpus once you're done?

Perplexity

- Information theoretic metric
  - Useful in measuring how well a grammar or language model (LM) models a natural language or a corpus

Perplexity: A function of the probability that a language model assigns to the test corpus.

$$P(W) = P(w_1,w_2...w_N)^{-1/N}$$

What perplexity does a LM(1) assign to the sentences of a test corpus, compared to another LM(2)?

Some Useful Empirical Observations

- A small number of events occur with high frequency
- A large number of events occur with low frequency
- You can quickly collect statistics on the high frequency events
- You might have to wait an arbitrarily long time to get valid statistics on low frequency events
- Some of the zeroes in the table are really zeros But others are simply low frequency events you haven't seen yet. How to address?

Smoothing

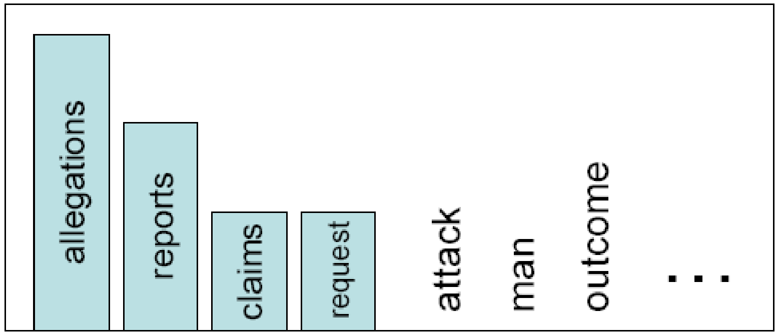
- Words follow a Zipfian distribution
  - Small number of words occur very frequently
  - A large number are seen only once
  - Zipf's law: a word's frequency is approximately inversely proportional to its rank in the word distribution list
- Zero probabilities on one bigram cause a zero probability on the entire sentence

Smoothing is like Robin Hood: Steal from the rich and give to the poor (in probability mass)

- We often want to make predictions from sparse statistics:

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

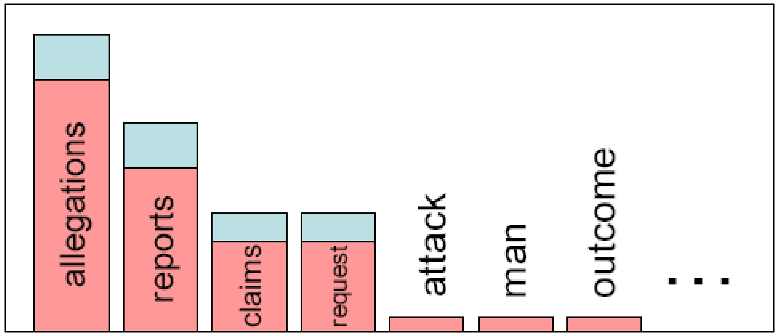
7 total



- Smoothing flattens spiky distributions so they generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other

7 total



- Very important all over NLP, but easy to do badly!

Slide from Dan Klein

Smoothing Techniques

- Every n-gram training matrix is sparse, even for very large corpora
  - Zipf’s law: a word’s frequency is approximately inversely proportional to its rank in the word distribution list
- Solution: estimate the likelihood of unseen n-grams
- Problems: how do you adjust the rest of the corpus to accommodate these ‘phantom’ n-grams?

Smoothing Methods

- Add-one smoothing (easy, but inaccurate)
  - Add 1 to every word count (Note: this is type)
  - Increment normalization factor by Vocabulary size: N (tokens) + *V (types)* :

$$p_i^* = \frac{c_i + 1}{N + V}$$

- Backoff models
  - When a count for an n-gram is 0, back off to the count for the (n-1)-gram
  - These can be weighted
- Class-based smoothing
  - For certain types of n-grams, back off to the count of its syntactic class
  - E.g., Count ProperNouns in place of names (e.g., Obama)
- Good-Turing
  - Re-estimate amount of probability mass for zero (or low count) ngrams by looking at ngrams with higher counts
  - Estimate

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

Summary

- N-gram probabilities can be used to *estimate* the likelihood
  - Of a word occurring in a context (N-1)
  - Of a sentence occurring at all
- Smoothing techniques deal with problems of unseen words in corpus
- Entropy and perplexity can be used to evaluate the information content of a language and the goodness of fit of a LM or grammar
- Read Ch. 5 on word classes and pos

Questions?

Homework 0