Ngram Review

October13, 2017

Professor Meteer

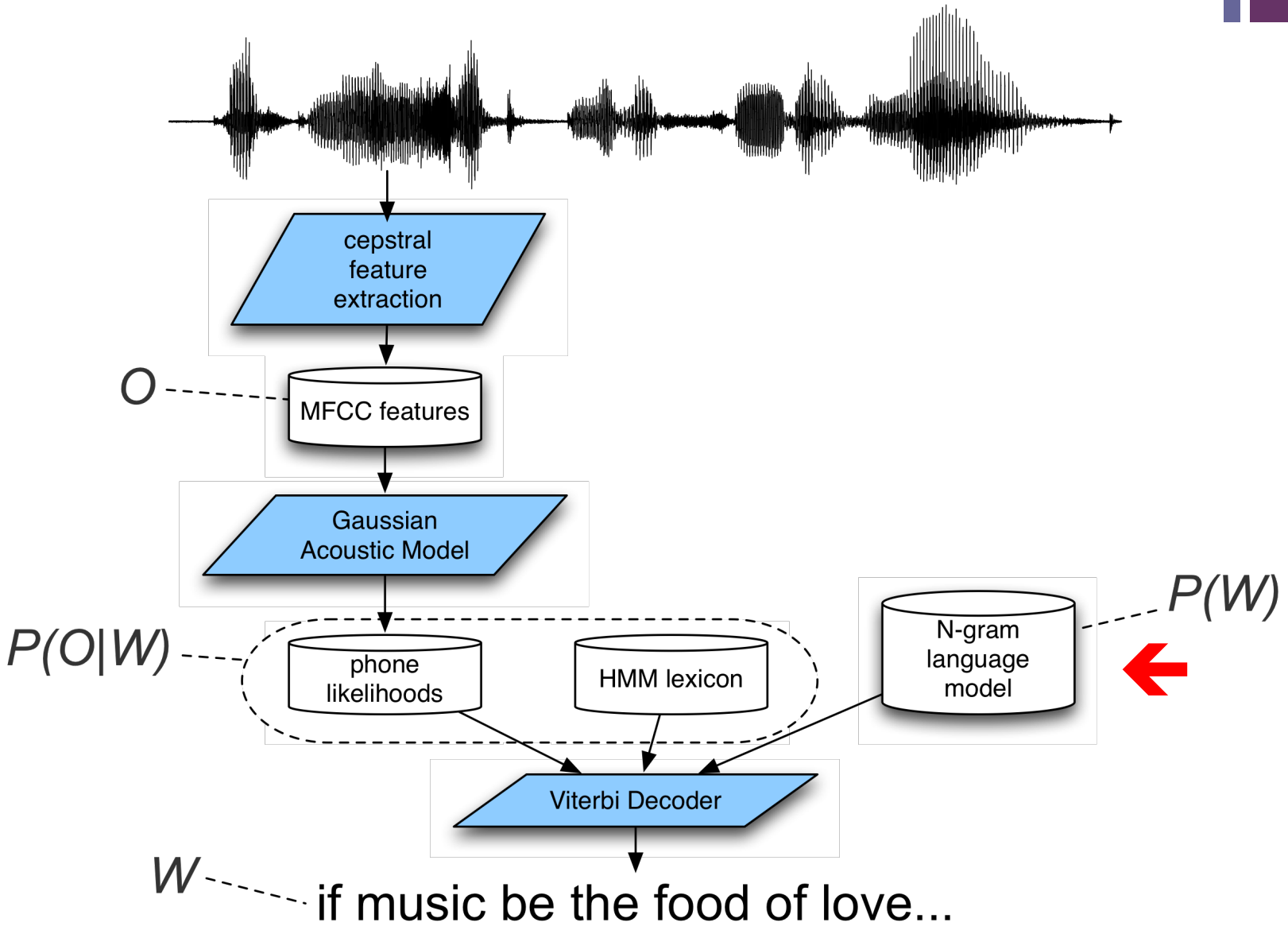
CS 136 Lecture 10 Language Modeling

Thanks to Dan Jurafsky for these slides

ASR components

- Feature Extraction, MFCCs, start of Acoustic
- HMMs, the Forward and Viterbi algorithms
- Baum-Welch (Forward-Backward)
- Acoustic Modeling and GMMs
- N-grams and Language Modeling
- Search and Advanced Decoding
- Dealing with Variation

Speech Recognition Architecture



Thanks to Dan Jurafsky for these slides

“Prior” probability of word sequence

- The ***language model*** captures the knowledge we have about what words to expect based on the ***context***
  - “Grammar” creates a finite state model of all (and only) possible sequences of words.
  - “Statistical Language Model” (SLM) encodes the probability of sequences of words based on counts from data.
- The ***context*** is both the domain and the immediate previous context
  - Domain: Language modeling data should match the target recognition data
  - Immediate context: Previous “n” words (usually 3-4)

+ Language Modeling

- We want to compute
  - $P(w_1,w_2,w_3,w_4,w_5 \ldots w_n) = P(W)$
  - = the probability of a sequence
- Alternatively we want to compute
  - $P(w_5|w_1,w_2,w_3,w_4)$
  - =the probability of a word given some previous words
- The model that computes
  - P(W) or
  - $P(w_n|w_1,w_2 \ldots w_{n-1})$
- We can model the word prediction task as the ability to assess the conditional probability of a word given the previous words in the sequence
  - $P(w_n|w_1,w_2 \ldots w_{n-1})$

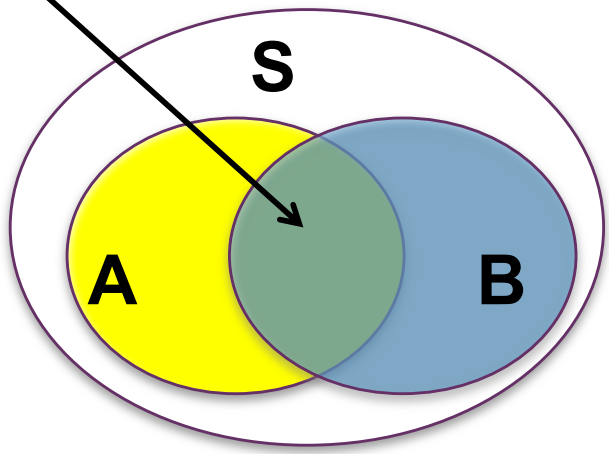
+ Conditional Probability

- Given an experiment, a corresponding sample space S, and a probability law
- Suppose we know that the outcome is within some given event B
- We want to quantify the likelihood that the outcome also belongs to some other given event A.
- We need a new probability law that gives us the conditional probability of A given B P(A|B)

Conditional Probability

- Let A and B be events
- p(B|A) = the probability of event B occurring given event A occurs
- Definition: $p(B|A) = p(A \cap B) / p(A)$
- So for LM:
  - $P(w_n|w_1,w_2 \ldots w_{n-1})$
  - $= P(w_1,w_2 \ldots w_n) / P(w_1,w_2 \ldots w_{n-1})$
- As in
  - P(*the* | *its water is so transparent that*)

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



Very Easy Estimate

- How to estimate?
  - P(the | its water is so transparent that) =

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

- According to Google those counts are 5/9
  - Unfortunately... 2 of those were to these slides... So maybe it's really 3/7
  - In any case, that's not terribly convincing due to the small numbers involved.
  - (actually, it's 95,800 / 103,000 or .95)

+ Language Modeling

- Unfortunately, for most sequences and for most text collections we won't get good estimates from this method.
  - What we're likely to get is 0. Or worse 0/0.
- Clearly, we'll have to be a little more clever.
  - Let's use the chain rule of probability
  - And a particularly useful independence assumption.

The Chain Rule

- Recall the definition of conditional probabilities

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

- For sequences...
  - P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)
- In general
  - $P(x_1,x_2,x_3,\ldots x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1\ldots x_{n-1})$

P(its water was so transparent)=

P(its)*

P(water|its)*

P(was|its water)*

P(so|its water was)*

P(transparent|its water was so)

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

+ Need the Independence Assumption

- There are still a lot of possible sentences
  - In general, we'll never be able to get enough data to compute the statistics for those longer prefixes
  - Same problem we had for the strings themselves
- Make the simplifying assumption
  - P(*the* | *its water is so transparent that*) =

    P(the | that)
- That is, the probability in question is independent of its earlier history.

Estimating Bigram Probabilities

- Markov Assumption
  - So for each component in the product replace with the approximation (assuming a prefix of N)

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

  - Bigram version

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-1})$$

- The Maximum Likelihood Estimate (MLE)

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

$$\frac{\text{full sequence}}{\text{context}}$$

Maximum Likelihood Estimates

- The maximum likelihood estimate of some parameter of a model M from a training set T
  - Is the estimate that maximizes the likelihood of the training set T given the model M
- Suppose the word Chinese occurs 400 times in a corpus of a million words (Brown corpus)
- What is the probability that a random word from some other text from the same distribution will be "Chinese"
- MLE estimate is 400/1000000 = .004
  - This may be a bad estimate for some other corpus
- But it is the **estimate** that makes it **most likely** that "Chinese" will occur 400 times in a million word corpus.

Berkeley Restaurant Project

- Data collected to create a language model for asking questions about restaurants near Berkeley

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

- Out of 9222 sentences
  - Eg. “I want” occurred 827 times

	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

Bigram Probabilities

- Divide bigram counts by prefix unigram counts to get probabilities.

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

	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

Kinds of Knowledge

- As crude as they are, *N*-gram probabilities capture a range of interesting facts about language.

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

World knowledge

- P(to|want) = .66
- P(eat | to) = .28

Syntax

- P(food | to) = 0
- P(want | spend) = 0

Discourse

- P (i | <s>) = .25

What to count?

- Each word?
- Ums and Uhs?
- Partial words?
- “polywords”? Classes of words?
- What about languages
  - With lots of inflections? (like Russian)
  - With no word boundaries (like Chinese)
  - With lots of compounding (like German)

Example from Switchboard

- A.1: Uh, do you have a pet Randy?
- B.2: Uh, yeah, currently we have a poodle.
- A.3: A poodle, miniature or, uh, full size?
- B.4: Yeah, uh, it's, uh miniature.
- A.5: Uh-huh.
- B.6: Yeah.
- A.7: I read somewhere that, the poodles is one of the, the most intelligent dogs, uh, around.
- B.8: Well, um, I wouldn't, uh, I definitely wouldn't dispute that, it, it's actually my wife's dog, uh, I, I became part owner six months ago when we got married, but, uh, it, uh, definitely responds to, uh, to authority and, I've had dogs in the past and, uh, it seems, it seems to, uh, respond real well, it, it - she's, she's picked up a lot of things, uh, just, just by, uh, teaching by force, I guess is what I'd like to say.
- A.9: Oh, uh-huh. So, you, you've only known the dog, wh-, how long did you say.

Shannon's Method

- Assigning probabilities to sentences is all well and good, but it's not terribly illuminating . A more interesting task is to turn the model around and use it to generate random sentences that are *like* the sentences from which the model was derived.
- Generally attributed to

  Claude Shannon.

+ Generating Shakespeare

- Unigrams
  - To him swallowed confess hear both. Which. Of save on trial for are ay device and rote life have c
  - Hill he late speaks; or! A more or legless first you enter
- Bigrams
  - What means, sir. I confess she? Then all sorts, he is trim, captain.
  - Why doest stand forth they canopy, forsooth he is this palpable hit the King Henry. Live king. Follow.
- Trigrams
  - Sweet prince, Falstaff shall die. Harry of Monmouths grave
  - This shall forbid it should be branded, if renown made it empty
- Quadrigrams
  - 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.

Unknown Words

- But once we start looking at test data, we'll run into words that we haven't seen before (pretty much regardless of how much training data you have.)
- With an Open Vocabulary task
  - Create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L, of size V
      - From a dictionary or
      - A subset of terms from the training set
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we count that like a normal word
  - At test time
    - Use UNK counts for any word not in training

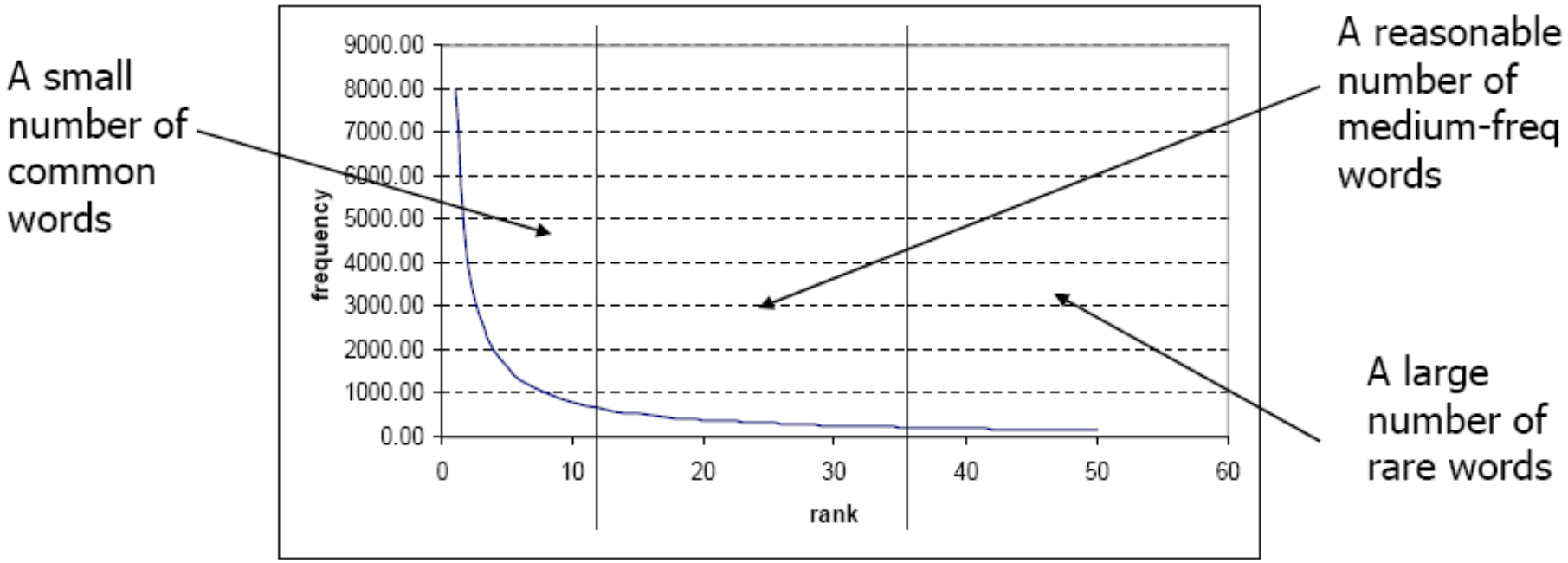
What to do about Zero Counts

- Back to Shakespeare
  - Recall that 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)
  - Does that mean that any sentence that contains one of those bigrams should have a probability of 0?

+ Zipf's Law

- Given the frequency *f* of a word and its rank *r* in the list of words ordered: by their frequencies:

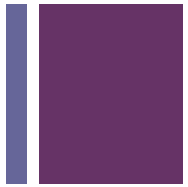
$f \propto 1/r$ or $f \times r = k$ for a constant k



Sparse Data Problem

- MLE is in general unsuitable for statistical inference in NLP because small parameters are hard to estimate.
- The problem is the sparseness of our data (even with the large corpus).
  - The vast majority of words are very uncommon
  - longer *n-grams involving them are thus much rarer*
- The MLE assigns a zero probability to unseen events
  - **Bad …**because the probability of the whole sequences will be zero
    - computed by multiplying the probabilities of subparts

+ Solution

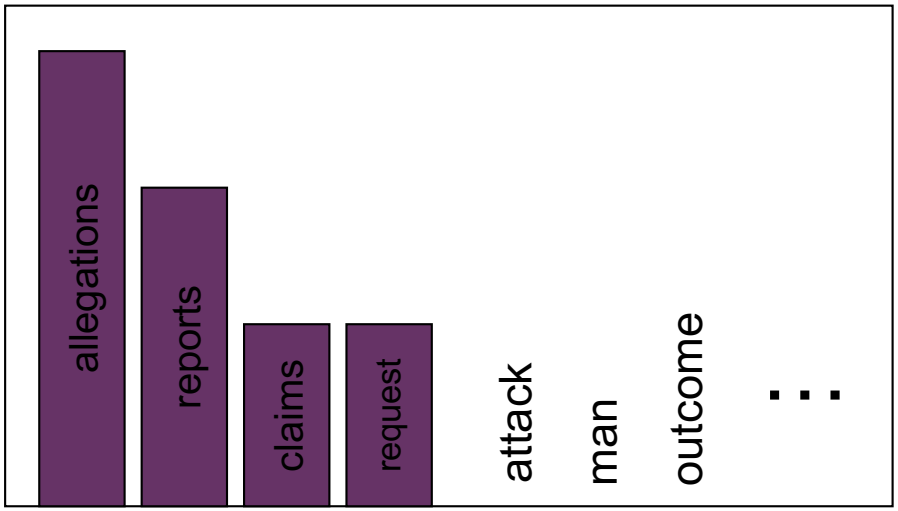


- How do you handle unseen n-grams?
  - Smoothing
    - Use some of the probability mass to cover unseen events
  - Backoff
    - Use counts from a smaller context
  - Interpolation
    - Combine multiple sources of information appropriately weighted
- Try to differentiate cases
  - Some of those zeros are really zeros...
    - Things that really can’t or shouldn’t happen.
  - Some of them are just rare events.
    - If the training corpus had been a little bigger they would have had a count (probably a count of 1!).

The intuition of smoothing (from Dan Klein)

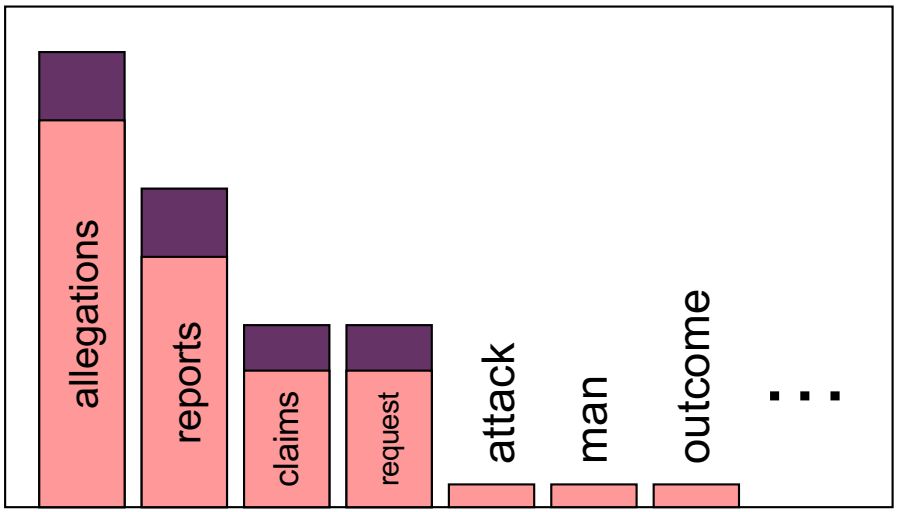
- When we have sparse statistics:

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



Laplace Smoothing

- Also called add-one smoothing
- Just add one to all the counts!
- Very simple

c_i: Counts for word I
N: Number of words
V: Size of the vocabulary

- MLE estimate: $$P(w_i) = \frac{c_i}{N}$$
- Laplace estimate: $$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$
- Reconstructed counts: $$c_i^* = (c_i + 1)\frac{N}{N + V}$$

Additive vs. Discounting approaches

- Problem: LaPlace is additive: adds 1 to everything
  - Gives too much probability mass to unseen n-grams
  - For sparse sets of data over large vocabularies, such as n-grams, Laplace's law actually gives far too much of the probability space to unseen events.
  - Can we smooth more usefully?
- Discounting (absolute discounting)
  - Subtracts ε from everything
  - Distributes ε across the unseen events

Better Smoothing

- Intuition used by many smoothing algorithms
  - Good-Turing
  - Kneser-Ney
  - Witten-Bell

- Is to use the count of things we've seen once to help estimate the count of things we've never seen

Backoff and Interpolation

- Smaller context can be a useful source of knowledge
- If we are estimating:
  - trigram p(z|x,y)
  - but count(xyz) is zero
- Use info from:
  - Bigram p(z|y)
- Or even:
  - Unigram p(z)
- How to combine this trigram, bigram, unigram info in a valid fashion?

Backoff Vs. Interpolation

- **Backoff**: use trigram if you have it, otherwise bigram, otherwise unigram
- **Interpolation**: mix all three (or other sources of knowledge)

+ Katz Backoff N-gram model

- If we've seen the n-gram, use it
  - But "discount it" by a normalizing factor
  - Have to account for "borrowing" for other unseen n-grams
- Otherwise: Recursively back off the the (N-1)-gram until there are some counts

$$P_{\text{katz}}(z|x,y) = \begin{cases} P^*(z|x,y), & \text{if } C(x,y,z) > 0 \\ \alpha(x,y)P_{\text{katz}}(z|y), & \text{else if } C(x,y) > 0 \\ P^*(z), & \text{otherwise.} \end{cases}$$

$$P_{\text{katz}}(z|y) = \begin{cases} P^*(z|y), & \text{if } C(y,z) > 0 \\ \alpha(y)P^*(z), & \text{otherwise.} \end{cases}$$

Thanks to Dan Jurafsky for these slides

Interpolation

- Simple interpolation

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

$$\sum_i \lambda_i = 1$$

- Lambdas conditional on context:

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

Smoothing: Kneser-Ney

P(Francisco | eggplant) vs P(stew | eggplant)

- "Francisco" is common, so backoff, interpolated methods say it is likely
- But it only occurs in context of "San"
- "Stew" is common, and in many contexts
- Weight backoff by number of contexts word occurs in
  - C = number of different Contexts
  - D = absolute discount (see textbook)

$$P_{IKN}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i) - D}{C(w_{i-1})} + \beta(w_i)\frac{|\{w_{i-1} : C(w_{i-1}w_i) > 0\}|}{\sum_{wi} |\{w_{i-1} : C(w_{i-1}w_i) > 0\}|}$$

+ Evaluating N-Gram Models

- Best evaluation for a language model
  - Put model A into an application
    - For example, a speech recognizer
  - Evaluate the performance of the application with model A
  - Put model B into the application and evaluate
  - Compare performance of the application with the two models
  - Extrinsic evaluation

Difficulty of extrinsic (in-vivo) evaluation of N-gram models

- Extrinsic evaluation
  - This is really time-consuming
  - Can take days to run an experiment
- So
  - As a temporary solution, in order to run experiments
  - To evaluate N-grams we often use an intrinsic evaluation, an approximation called perplexity
  - But perplexity is a poor approximation unless the test data looks just like the training data
  - So is generally only useful in pilot experiments (generally is not sufficient to publish)
- But is helpful to think about.

+ Evaluation

- Standard method
  - Train parameters of our model on a training set.
  - Look at the models performance on some new data
    - This is exactly what happens in the real world; we want to know how our model performs on data we haven't seen
  - So use a test set. A dataset which is different than our training set, but is drawn from the same source
  - Then we need an evaluation metric to tell us how well our model is doing on the test set.
    - One such metric is **perplexity**

Intuition of Perplexity

- 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 ____

  - Unigrams are terrible at this game. (Why?)

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

….

and 1e-100

- A better model of a text
  - is one which assigns a higher probability to the word that actually occurs
- Ask a speech recognizer to recognize digits: “0, 1, 2, 3, 4, 5, 6, 7, 8, 9” – easy – perplexity 10
- Perplexity is weighted equivalent branching factor.

Example: Linguistic Segmentation

- Acoustic segmentation
  - I'm not sure how many active volcanoes there are now and and what the amount of material that they do
  - uh put into the atmosphere
  - I think probably the greatest cause is uh
  - vehicles
  - especially around cities
- Linguistic segmentation
  - I'm not sure how many active volcanoes there are now and and what the amount of material that they do uh put into the atmosphere
  - I think probably the greatest cause is uh vehicles especially around cities

Compare perplexity

- Build three models

Test		Training	
	Acoustic Seg	Ling Seg	No Seg
Acoustic Seg	105	111	
Ling Seg	89	78	
No Seg	163	174	130

Small enough

- Real language models are often huge
- 5-gram models typically larger than the training data
- Use count-cutoffs (eliminate parameters with fewer counts) or, better
- Use Stolcke pruning – finds counts that contribute least to perplexity reduction,
  - P(City | New York) ≈ P(City | York)
  - P(Friday | God it's) ≠ P(Friday | it's)
- Remember, Kneser-Ney helped most when lots of 1 counts

N-gram versus smoothing algorithm

n-gram	Katz	Kneser-Ney
2	134	132
3	80	74
4	75	65
5	78	62

Overview (from Microsoft Tutorial)

- Caching
- Skipping
- Clustering
- Sentence-mixture models
- Structured language models
- Tools
- More on the author, Josh Goodman
  http://research.microsoft.com/en-us/um/people/joshuago/icmldescription.htm

+ Sentence Mixture Models

- Lots of different sentence types:
  - Numbers (The Dow rose one hundred seventy three points)
  - Quotations (Officials said “quote we deny all wrong doing ”quote)
  - Mergers (AOL and Time Warner, in an attempt to control the media and the internet, will merge)
- Model each sentence type separately

Sentence Mixture Models

- Roll a die to pick sentence type, s_k

with probability λ_k

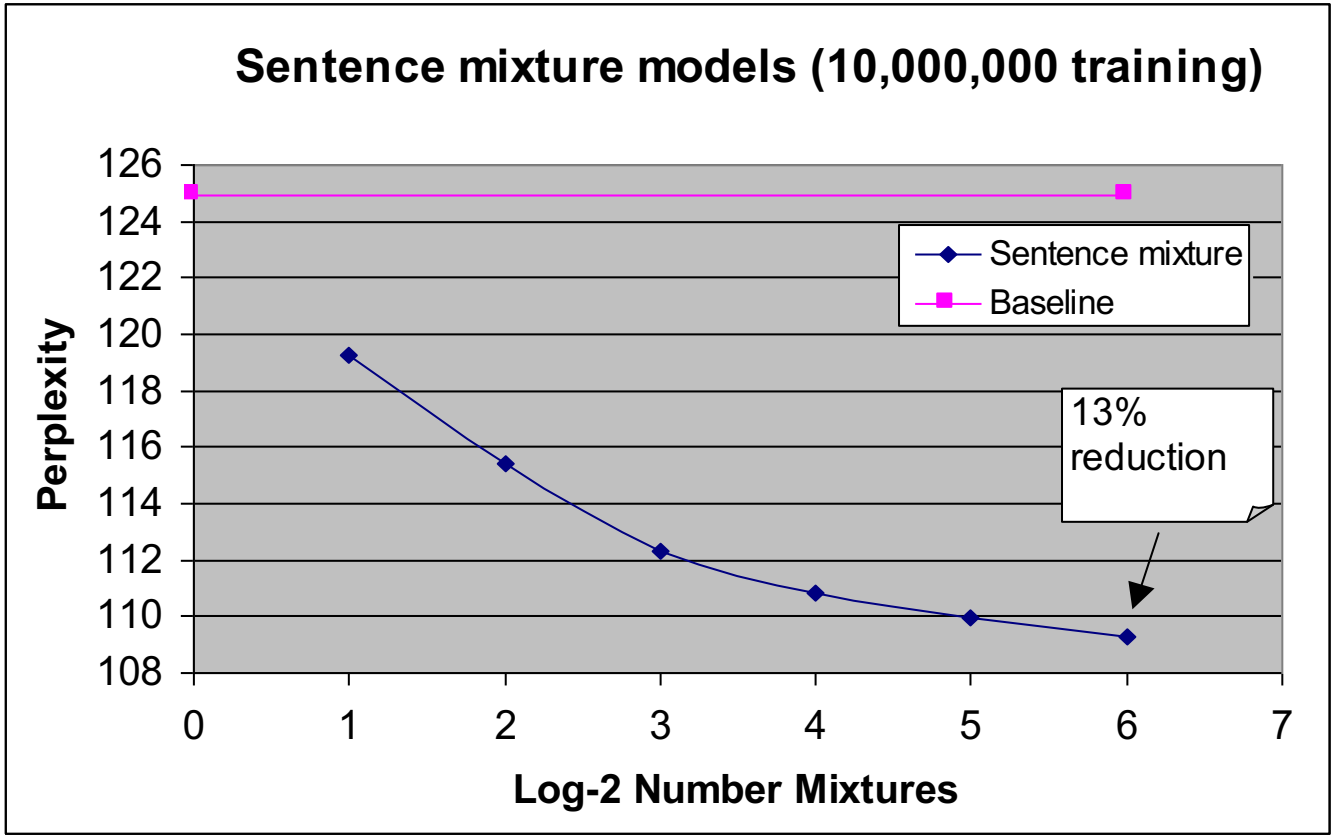
- Probability of sentence, given s_k

$$\prod_{i=1}^{n} P(w_i \mid w_{i-2} w_{i-1} s_k)$$

- Probability of sentence across types:

$$\sum_{k=1}^{m} \lambda_k \prod_{i=1}^{n} P(w_i \mid w_{i-2} w_{i-1} s_k)$$

Sentence Mixture Results



Topic Examples - 0 (Mergers and acquisitions)

- JOHN BLAIR & COMPANY IS CLOSE TO AN AGREEMENT TO SELL ITS T. V. STATION ADVERTISING REPRESENTATION OPERATION AND PROGRAM PRODUCTION UNIT TO AN INVESTOR GROUP LED BY JAMES H. ROSENFIELD ,COMMA A FORMER C. B. S. INCORPORATED EXECUTIVE ,COMMA INDUSTRY SOURCES SAID .PERIOD
- INDUSTRY SOURCES PUT THE VALUE OF THE PROPOSED ACQUISITION AT MORE THAN ONE HUNDRED MILLION DOLLARS .PERIOD
- JOHN BLAIR WAS ACQUIRED LAST YEAR BY RELIANCE CAPITAL GROUP INCORPORATED ,COMMA WHICH HAS BEEN DIVESTING ITSELF OF JOHN BLAIR'S MAJOR ASSETS .PERIOD
- JOHN BLAIR REPRESENTS ABOUT ONE HUNDRED THIRTY LOCAL TELEVISION STATIONS IN THE PLACEMENT OF NATIONAL AND OTHER ADVERTISING .PERIOD
- MR. ROSENFIELD STEPPED DOWN AS A SENIOR EXECUTIVE VICE PRESIDENT OF C. B. S. BROADCASTING IN DECEMBER NINETEEN EIGHTY FIVE UNDER A C. B. S. EARLY RETIREMENT PROGRAM .PERIOD

Topic Examples - 2 (Numbers)

- SOUTH KOREA POSTED A SURPLUS ON ITS CURRENT ACCOUNT OF FOUR HUNDRED NINETEEN MILLION DOLLARS IN FEBRUARY ,COMMA IN CONTRAST TO A DEFICIT OF ONE HUNDRED TWELVE MILLION DOLLARS A YEAR EARLIER ,COMMA THE GOVERNMENT SAID .PERIOD
- THE CURRENT ACCOUNT COMPRISES TRADE IN GOODS AND SERVICES AND SOME UNILATERAL TRANSFERS .PERIOD
- COMMERCIAL -HYPHEN VEHICLE SALES IN ITALY ROSE ELEVEN .POINT FOUR %PERCENT IN FEBRUARY FROM A YEAR EARLIER ,COMMA TO EIGHT THOUSAND ,COMMA EIGHT HUNDRED FORTY EIGHT UNITS ,COMMA ACCORDING TO PROVISIONAL FIGURES FROM THE ITALIAN ASSOCIATION OF AUTO MAKERS .PERIOD
- INDUSTRIAL PRODUCTION IN ITALY DECLINED THREE .POINT FOUR %PERCENT IN JANUARY FROM A YEAR EARLIER ,COMMA THE GOVERNMENT SAID .PERIOD

Topic Examples – 3 (quotations)

- NEITHER MR. ROSENFIELD NOR OFFICIALS OF JOHN BLAIR COULD BE REACHED FOR COMMENT .PERIOD
- THE AGENCY SAID THERE IS "DOUBLE-QUOTE SOME INDICATION OF AN UPTURN "DOUBLE-QUOTE IN THE RECENT IRREGULAR PATTERN OF SHIPMENTS ,COMMA FOLLOWING THE GENERALLY DOWNWARD TREND RECORDED DURING THE FIRST HALF OF NINETEEN EIGHTY SIX .PERIOD
- THE COMPANY SAID IT ISN'T AWARE OF ANY TAKEOVER INTEREST .PERIOD
- THE SALE INCLUDES THE RIGHTS TO GERMAINE MONTEIL IN NORTH AND SOUTH AMERICA AND IN THE FAR EAST ,COMMA AS WELL AS THE WORLDWIDE RIGHTS TO THE DIANE VON FURSTENBERG COSMETICS AND FRAGRANCE LINES AND U. S. DISTRIBUTION RIGHTS TO LANCASTER BEAUTY PRODUCTS .PERIOD
- BUT THE COMPANY WOULDN'T ELABORATE .PERIOD

Reality: Text normalization

- What about “$3,100,000” → convert to “Three million one hundred thousand dollars”, etc.
- Need to do this for dates, numbers, maybe abbreviations.
- Some text-normalization tools come with Wall Street Journal corpus, from LDC (Linguistic Data Consortium)
- Not much available
- Write your own (use Perl!)

Lattices in Kaldi

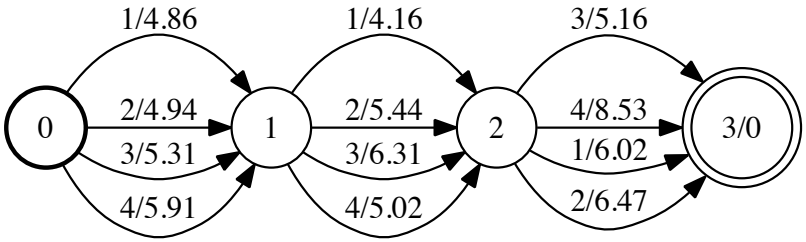
- Representation of the alternative word-sequences that are "sufficiently likely" for a particular utterance
  - The lattice should have a path for every word sequence within α of the best-scoring one.
  - The scores and alignments in the lattice should be accurate.
  - The lattice should not contain duplicate paths with the same word sequence.
- We begin with a Weighted Finite State Transducer

  HCLG=min(det(H∘C∘L∘G))

  - H: HMM
  - C: Context dependent phonemes
  - L: Lexicon
  - G: Grammar

WFSTs and Decoding

- An input utterance U is a set of feature vectors of length T
  - U: Utterance is a WFSA with T+1 states
  - One arc for every combination time +state



- The search graph is defined as

$$S \equiv \mathbf{U} \circ \mathbf{HCLG}$$

  - S has approximately T+1 times as many states as HCLG
- Decoding is finding best path through S
  - In reality, searching through a subset of S that has been pruned

Operations on lattices

- Pruning lattices
  - Use a specified beam to remove states and arcs that are not on a path sufficiently close to the cost of the best path through the lattice.
- Computing the best path
  - outputs the corresponding input-symbol sequence (alignment) and output-symbol sequence (transcription) of the best path
- Computing the N-best hypotheses
  - Outputs a lattice with a new start state with (up to) n arcs, each starting a separate path that is within the top N scoring paths

Language model rescoring

- Lattice weights are a combination of language model + transition probabilities + pronunciation/silence probabilities.
- First need to subtract the original LM probabilities then add the new LM probabilities
  - lattice-lmrescore --lm-scale=-1.0 ark:in.lats G_old.fst ark:nolm.lats
  - lattice-lmrescore --lm-scale=1.0 ark:nolm.lats G_new.fst ark:out.lats
- NOTE: Lexicon has to be the same!