Statistical NLP
Fall 2015

Lecture 1:
Introduction & Language Modeling

Slav Petrov – Google

Many thanks to Dan Klein (UC Berkeley) for slides and class materials
Administrivia

http://cs.nyu.edu/courses/fall15/CSCI-GA.3033-008/

Statistical Natural Language Processing Fall 2015

Course#: CSCI-GA.3033-008
Instructor: Slav Petrov
Lecture: Tuesdays 5:10-7:00PM, Warren Weaver Hall Room 512
Mailing List: Piazza
Office hours: By appointment

Announcements:

21/04/15: Webpage is up.
28/08/15: Update the webpage and syllabus.
Announcements

- **Communication:**
  - Announcements and public discussion: Piazza! 
    https://piazza.com/nyu/fall2015/csciga3033008/home
  - My email is petrov@cs.nyu.edu, but use Piazza whenever possible.

- **Computing Resources**
  - You might want more compute power than your laptop
  - Experiments will take minutes to hours, with efficient code
  - Recommendation: start assignments early

- **Questions?**
Course Details

- **Books:**
  - Jurafsky and Martin, Speech and Language Processing, 2 Ed
  - Manning and Schuetze, Foundations of Statistical NLP (available online)

- **Additional readings from recent papers**

- **Prerequisites:**
  - Solid math background
  - Solid skills in Java or equivalent
  - Deep interest in language
Work and Grading

- **Class participation (30%)**
  - Participate in class!
  - 1 substantial question and 1 substantial answer to Piazza every month

- **Five programming assignments (40%)**
  - Individual, write-ups
  - 5 points for beating baseline
  - 1 bonus point for first submission to beat baseline
  - 2 bonus points for best submission
  - 1 bonus point for second best submission

- **Late policy:**
  - You get 7 late days to use at your discretion (no more than 5 per assignment)
  - After that, you lose 10% per day

- **Final project (30%)**
  - Can be done in groups
AI: Where do we stand?

Hollywood

1980
R2D2
Rule based approaches

1990
KITT
Early statistical approaches

2000
Wall-E
Modern statistical approaches

2010
Self-Driving Cars

2020
Phone Assistants
What is NLP?

Fundamental goal: deep understand of broad language
  - Not just string processing or keyword matching!

End systems that we want to build:
  - Simple: spelling correction, text categorization…
  - Complex: speech recognition, machine translation, information extraction, dialog interfaces, question answering…
  - Unknown: human-level comprehension (is this just NLP?)
Speech Systems

- **Automatic Speech Recognition (ASR)**
  - Audio in, text out
  - State-of-the-art (SOTA): 0.3% error for digit strings, 5% dictation, 50%+ TV

- **Text to Speech (TTS)**
  - Text in, audio out
  - SOTA: totally intelligible (if sometimes unnatural)
Information Extraction

- Unstructured text to database entries

New York Times Co. named Russell T. Lewis, 45, president and general manager of its flagship New York Times newspaper, responsible for all business-side activities. He was executive vice president and deputy general manager. He succeeds Lance R. Primis, who in September was named president and chief operating officer of the parent.

<table>
<thead>
<tr>
<th>Person</th>
<th>Company</th>
<th>Post</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Russell T. Lewis</td>
<td>New York Times newspaper</td>
<td>president and general manager</td>
<td>start</td>
</tr>
<tr>
<td>Russell T. Lewis</td>
<td>New York Times newspaper</td>
<td>executive vice president</td>
<td>end</td>
</tr>
<tr>
<td>Lance R. Primis</td>
<td>New York Times Co.</td>
<td>president and CEO</td>
<td>start</td>
</tr>
</tbody>
</table>

- SOTA: perhaps 80% accuracy for multi-sentence templates, 90% + for single easy fields
- But remember: information is redundant!
Question Answering

- **Question Answering:**
  - More than search
  - Ask general comprehension questions of a document collection
  - Can be really easy: “What’s the capital of Wyoming?”
  - Can be harder: “How many US states’ capitals are also their largest cities?”
  - Can be open ended: “What are the main issues in the global warming debate?”

- **SOTA:** Can do factoids, even when text isn’t a perfect match
who is the chef at The French Laundry

Thomas Keller

“The French Laundry is a French restaurant located in Yountville, California, in the Napa Valley. The chef and owner of the French Laundry is Thomas Keller. The restaurant building dates from 1990, and is in the National Register of Historic Places.”

what are its hours

5:30 – 9:15 pm

The French Laundry, Thursday hours

what books has he written

Thomas Keller

Books

- The French Laundry Cookbook 1999
- An Odfrm And Mc-cdma Primer 2006
- Single and Multicarrier Modulation 2000
- Angels: The Lifting of the Veil 1994
- Computers for Managing Information Stu... 1988
Summarization

- Condensing documents
  - Single or multiple
  - Extractive or synthetic
  - Aggregative or representative
  - Even just shortening sentences
- Very context-dependent!
- An example of analysis with generation
Machine Translation

- Translate text from one language to another
- Recombines fragments of example translations
- Challenges:
  - What fragments?  [learning to translate]
  - How to make efficient?  [fast translation search]
  - Fluency (next class) vs fidelity (later)
"The rock was still wet. The animal was glistening, like it was still swimming," recalls Hou Xianguang. Hou discovered the unusual fossil while surveying rocks as a paleontology graduate student in 1984, near the Chinese town of Chengjiang. "My teachers always talked about the Burgess Shale animals. It looked like one of them. My hands began to shake." Hou had indeed found a Naraoia like those from Canada. However, Hou's animal was 15 million years older than its Canadian relatives.

It can be inferred that Hou Xianguang's "hands began to shake", because he was:

(A) afraid that he might lose the fossil
(B) worried about the implications of his finding
(C) concerned that he might not get credit for his work
(D) uncertain about the authenticity of the fossil
(E) excited about the magnitude of his discovery
Why is NLP Hard?

- The core problems:
  - Ambiguity
  - Scale
  - Sparsity
  - Unmodeled Variables
Problem: Ambiguities

- Headlines:
  - Hospitals Are Sued by 7 Foot Doctors
  - Iraqi Head Seeks Arms
  - Enraged Cow Injures Farmer with Ax
  - Ban on Nude Dancing on Governor’s Desk
  - Stolen Painting Found by Tree
  - Teacher Strikes Idle Kids
  - Kids Make Nutritious Snacks
  - Local HS Dropouts Cut in Half

- Why are these funny?
Hurricane Emily howled toward Mexico's Caribbean coast on Sunday packing 135 mph winds and torrential rain and causing panic in Cancun, where frightened tourists squeezed into musty shelters.

- SOTA: ~90% accurate for many languages when given many training examples, some progress in analyzing languages given few or no examples
Semantic Ambiguity

- NLP is much more than syntax!
- Even correct tree structured syntactic analyses don’t fully nail down the meaning

*John’s boss said he was doing better*

*Every morning someone’s alarm clock wakes me up*

- In general, every level of linguistic structure comes with its own ambiguities…
Problem: Scale

- People *did* know that language was ambiguous!
  - …but they hoped that all interpretations would be “good” ones (or ruled out pragmatically)
  - …they didn’t realize how bad it would be
Corpora

- A corpus is a collection of text
  - Annotated in some way: supervised learning
  - Sometimes just lots of text without any annotations: unsupervised learning
  - Balanced vs. uniform corpora

- Examples
  - Newswire collections: 500M+ words
  - Brown corpus: 1M words of tagged “balanced” text
  - Penn Treebank: 1M words of parsed WSJ
  - Canadian Hansards: 10M+ words of aligned French / English sentences
  - The Web: billions of words of who knows what
Supervised vs. Unsupervised Learning

- In most NLP work, supervised methods are necessary for the state of the art.

- But unsupervised methods are the promised land.

- We will cover both types of methods.
Language isn’t Adversarial

- One nice thing: we know NLP can be done!

- Language isn’t adversarial:
  - It’s produced with the intent of being understood
  - With some understanding of language, you can often tell what knowledge sources are relevant

- But most variables go unmodeled
  - Some knowledge sources aren’t easily available (real-world knowledge, complex models of other people’s plans)
  - Some kinds of features are beyond our technical ability to model (especially cross-sentence correlations)
What is this Class?

Three aspects to the course:

- **Linguistic Issues**
  - What are the range of language phenomena?
  - What are the knowledge sources that let us disambiguate?
  - What representations are appropriate?
  - How do you know what to model and what not to model?

- **Statistical Modeling Methods**
  - Increasingly complex model structures
  - Learning and parameter estimation
  - Efficient inference: dynamic programming, search, sampling

- **Engineering Methods**
  - Issues of scale
  - Where the theory breaks down (and what to do about it)

- We’ll focus on what makes the problems hard, and what works in practice…
Class Requirements and Goals

- **Class requirements**
  - Uses a variety of skills / knowledge:
    - Probability and statistics
    - Basic linguistics background (ling101)
    - Decent coding skills (Java), knowledge of data structures
  - Most people are probably missing one of the above
  - You will often have to work on your own to fill the gaps

- **Class goals**
  - Learn the issues and techniques of statistical NLP
  - Build realistic tools used in NLP (language models, taggers, parsers, translation systems)
  - Be able to read current research papers in the field
  - See where the holes in the field still are!
The (Effective) NLP Cycle

- Pick a problem (usually some disambiguation)
- Get a lot of data (hopefully labeled, but often unlabeled)
- Build the simplest thing that could possibly work
- Repeat:
  - Examine the most common errors are
  - Figure out what information a human might use to avoid them
  - Modify the system to exploit that information
    - Feature engineering
    - Representation redesign
    - Different machine learning methods
- We’re going to do this over and over again
Some Disclaimers

- The purpose of this class is to train NLP researchers
  - Some people will put in a lot of time
  - There will be a lot of reading, some required, some not – you will have to be strategic about what reading enables your goals
  - There will be a lot of coding and running systems on substantial amounts of real data
  - There will be a lot of statistical modeling (though we do use a few basic techniques very heavily)
  - There will be discussion and questions in class that will push past what I present in lecture, and I’ll answer them
  - Not everything will be spelled out for you in the assignments

- Don’t say I didn’t warn you!
Outline of Topics

- **Words**
  - N-gram models and smoothing
  - Classification and clustering
- **Sequences**
  - Part-of-speech tagging
  - Word-alignments
- **Trees**
  - Syntax
  - Semantics
  - Machine translation
- **Sentiment Analysis**
- **Summarization**
- **Generative Models**
- **Discriminative Models**
- **Graphical Models**
- **Neural Networks**
Speech in a Slide

- Frequency gives pitch; amplitude gives volume

- Frequencies at each time slice processed into observation vectors
The Noisy-Channel Model

- We want to predict a sentence given acoustics:

\[ w^* = \arg \max_w P(w|a) \]

- The noisy channel approach:

\[ w^* = \arg \max_w P(w|a) \]

\[ = \arg \max_w P(a|w)P(w)/P(a) \]

\[ \propto \arg \max_w P(a|w)P(w) \]

Acoustic model: HMMs over word positions with mixtures of Gaussians as emissions

Language model: Distributions over sequences of words (sentences)
Acoustically Scored Hypotheses

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>the station signs are in deep in english</td>
<td>-14732</td>
</tr>
<tr>
<td>the stations signs are in deep in english</td>
<td>-14735</td>
</tr>
<tr>
<td>the station signs are in deep into english</td>
<td>-14739</td>
</tr>
<tr>
<td>the station 's signs are in deep in english</td>
<td>-14740</td>
</tr>
<tr>
<td>the station signs are in deep in the english</td>
<td>-14741</td>
</tr>
<tr>
<td>the station signs are indeed in english</td>
<td>-14757</td>
</tr>
<tr>
<td>the station 's signs are indeed in english</td>
<td>-14760</td>
</tr>
<tr>
<td>the station signs are indians in english</td>
<td>-14790</td>
</tr>
<tr>
<td>the station signs are indian in english</td>
<td>-14799</td>
</tr>
<tr>
<td>the stations signs are indians in english</td>
<td>-14807</td>
</tr>
<tr>
<td>the stations signs are indians and english</td>
<td>-14815</td>
</tr>
</tbody>
</table>
ASR System Components

Language Model

source $P(w)$

best $w$

decoder

Acoustic Model

channel $P(a|w)$

observed $a$

$\arg\max_w P(w|a) = \arg\max_w P(a|w)P(w)$
Translation: Codebreaking?

“Also knowing nothing official about, but having guessed and inferred considerable about, the powerful new mechanized methods in cryptography—methods which I believe succeed even when one does not know what language has been coded—one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: ‘This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.’”

Warren Weaver (1955:18, quoting a letter he wrote in 1947)
MT System Components

Language Model

source
P(e)

best
e

decoder

Translation Model

channel
P(f|e)

observed
f

argmax P(e|f) = argmax P(f|e)P(e)
Other Noisy-Channel Processes

- **Spelling Correction**
  
  \[ P(\text{words} \mid \text{characters}) \propto P(\text{words})P(\text{characters} \mid \text{words}) \]

- **Handwriting recognition**

  \[ P(\text{words} \mid \text{strokes}) \propto P(\text{words})P(\text{strokes} \mid \text{words}) \]

- **OCR**

  \[ P(\text{words} \mid \text{pixels}) \propto P(\text{words})P(\text{pixels} \mid \text{words}) \]

- **More…**
Probabilistic Language Models

- **Goal:** Assign useful probabilities $P(x)$ to sentences $x$
  - Input: many observations of training sentences $x$
  - Output: system capable of computing $P(x)$

- Probabilities should broadly indicate plausibility of sentences
  - $P(I \text{ saw a van}) \gg P(\text{eyes awe of an})$
  - *Not grammaticality:* $P(\text{artichokes intimidate zippers}) \approx 0$
  - In principle, “plausible” depends on the domain, context, speaker…

- **One option:** empirical distribution over training sentences?
  - Problem: doesn’t generalize (at all)

- **Two aspects of generalization**
  - Decomposition: break sentences into small pieces which can be recombined in new ways (conditional independence)
  - Smoothing: allow for the possibility of unseen pieces
N-Gram Model Decomposition

- Chain rule: break sentence probability down

\[ P(w_1 \ldots w_n) = \prod_i P(w_i | w_1 \ldots w_{i-1}) \]

- Impractical to condition on everything before
  - \( P(???) \mid \text{Turn to page 134 and look at the picture of the} \) ?

- N-gram models: assume each word depends only on a short linear history

\[ P(w_1 \ldots w_n) = \prod_i P(w_i | w_{i-k} \ldots w_{i-1}) \]

- Example:

\[
P(\text{please close the door}) =
\]

\[ P(\text{please} | \text{START}) P(\text{close} | \text{please}) \ldots P(\text{STOP} | \text{door}) \]
The parameters of an n-gram model:
- The actual conditional probability estimates, we’ll call them $\theta$
- Obvious estimate: relative frequency (maximum likelihood) estimate

$$\hat{P}(w|w_{-1}) = \frac{c(w_{-1}, w)}{\sum_{w'} c(w_{-1}, w')}$$

General approach
- Take a training set $X$ and a test set $X'$
- Compute an estimate $\theta$ from $X$
- Use it to assign probabilities to other sentences, such as those in $X'$

<table>
<thead>
<tr>
<th>Training Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>198015222 the first</td>
</tr>
<tr>
<td>194623024 the same</td>
</tr>
<tr>
<td>168504105 the following</td>
</tr>
<tr>
<td>158562063 the world</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>14112454 the door</td>
</tr>
<tr>
<td>23135851162 the *</td>
</tr>
</tbody>
</table>

$$\hat{P}(\text{door}|\text{the}) = \frac{14112454}{23135851162} = 0.0006$$
Higher Order N-grams?

Please close the door

---

Please close the first window on the left

---

198015222 the first
194623024 the same
168504105 the following
158562063 the world
...
14112454 the door
----------
23135851162 the *

197302 close the window
191125 close the door
152500 close the gap
116451 close the thread
87298 close the deal
----------
3785230 close the *

3380 please close the door
1601 please close the window
1164 please close the new
1159 please close the gate
900 please close the browser
----------
13951 please close the *
Unigram Models

- Simplest case: unigrams

\[
P(w_1 \ldots w_n) = \prod_i P(w_i)
\]

- Generative process: pick a word, pick a word, … until you pick STOP

- As a graphical model:

![Graphical model of unigrams]

- Examples:
  - [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]
  - []
  - [after, any, on, consistently, hospital, lake, of, of, other, and, factors, raised, analyst, too, allowed, mexico, never, consider, fall, bungled, davison, that, obtain, price, lines, the, to, sass, the, the, further, board, a, details, machinists, the, companies, which, rivals, an, because, longer, oakes, percent, a, they, three, edward, it, currier, an, within, in, three, wrote, is, you, s., longer, institute, dentistry, pay, however, said, possible, to, rooms, hiding, eggs, approximate, financial, canada, the, so, workers, advancers, half, between, nasdaq]
Bigram Models

- Big problem with unigrams: $P(\text{the the the the}) >> P(\text{I like ice cream})$!
- Condition on previous single word:

$$P(w_1 \ldots w_n) = \prod_{i} P(w_i | w_{i-1})$$

- Obvious that this should help – in probabilistic terms, we’re using weaker conditional independence assumptions (what’s the cost?)
- Any better?
  - [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]
  - [although, common, shares, rose, forty, six, point, four, hundred, dollars, from, thirty, seconds, at, the, greatest, play, disingenuous, to, be, reset, annually, the, buy, out, of, american, brands, vying, for, mr., womack, currently, sharedata, incorporated, believe, chemical, prices, undoubtedly, will, be, as, much, is, scheduled, to, conscientious, teaching]
  - [this, would, be, a, record, november]
Regular Languages?

- **N-gram models are (weighted) regular languages**
  - Many linguistic arguments that language isn’t regular.
    - Long-distance effects: “The computer which I had just put into the machine room on the fifth floor ____.” (crashed)
  - Recursive structure
  - Why CAN we often get away with n-gram models?

- **PCFG LM (later):**
  - [This, quarter, ‘s, surprisingly, independent, attack, paid, off, the, risk, involving, IRS, leaders, and, transportation, prices, .]
  - [It, could, be, announced, sometime, .]
  - [Mr., Toseland, believes, the, average, defense, economy, is, drafted, from, slightly, more, than, 12, stocks, .]
## More N-Gram Examples

<table>
<thead>
<tr>
<th>Unigram</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have</td>
<td></td>
</tr>
<tr>
<td>• Every enter now severally so, let</td>
<td></td>
</tr>
<tr>
<td>• Hill he late speaks; or! a more to leg less first you enter</td>
<td></td>
</tr>
<tr>
<td>• Are where exeunt and sighs have rise excellency took of. Sleep knave we. near; vile like</td>
<td></td>
</tr>
</tbody>
</table>
Measuring Model Quality

- The game isn’t to pound out fake sentences!
  - Obviously, generated sentences get “better” as we increase the model order
  - More precisely: using ML estimators, higher order is always better likelihood on train, but not test

- What we really want to know is:
  - Will our model prefer good sentences to bad ones?
  - Bad ≠ ungrammatical!
  - Bad ≈ unlikely
  - Bad = sentences that our acoustic model really likes but aren’t the correct answer
The Shannon Game:

- How well can we predict the next word?
  
  When I eat pizza, I wipe off the ____
  Many children are allergic to ____
  I saw a ____

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

“Entropy”: per-word test

log likelihood

$$H(X|\theta) = -\frac{1}{|X|} \sum_{x \in X} \log_2 P(x|\theta)$$

$$\sum_{x \in X} |x| \sum_{i} \log P(x_i|x_{i-1}, \theta)$$

<table>
<thead>
<tr>
<th>Word</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>grease</td>
<td>0.5</td>
</tr>
<tr>
<td>sauce</td>
<td>0.4</td>
</tr>
<tr>
<td>dust</td>
<td>0.05</td>
</tr>
<tr>
<td>mice</td>
<td>0.0001</td>
</tr>
<tr>
<td>the</td>
<td>1e-100</td>
</tr>
</tbody>
</table>

```
3516 wipe off the excess
1034 wipe off the dust
547 wipe off the sweat
518 wipe off the mouthpiece
...
120 wipe off the grease
0 wipe off the sauce
0 wipe off the mice
-------------
28048 wipe off the *
```
Measuring Model Quality

- **Problem with “entropy”:**
  - 0.1 bits of improvement doesn’t sound so good
  - “Solution”: perplexity

\[
\text{perp}(X, \theta) = 2^{H(X|\theta)}
\]

- Interpretation: average branching factor in model

- **Important notes:**
  - It’s easy to get bogus perplexities by having bogus probabilities that sum to more than one over their event spaces. 30% of you will do this on HW1.
  - Even though our models require a stop step, averages are per actual word, not per derivation step.
Measuring Model Quality

- Word Error Rate (WER)
  \[ \frac{\text{insertions} + \text{deletions} + \text{substitutions}}{\text{true sentence size}} \]

  Correct answer: Andy saw a part of the movie
  Recognizer output: And he saw apart of the movie

  WER: \( \frac{4}{7} = 57\% \)

- The “right” measure:
  - Task error driven
  - For speech recognition
  - For a specific recognizer!

- Common issue: intrinsic measures like perplexity are easier to use, but extrinsic ones are more credible
Problems with n-gram models:
- New words appear all the time:
  - Synaptitute
  - 132,701.03
  - multidisciplinarization
- New bigrams: even more often
- Trigrams or more – still worse!

Zipf’s Law
- Types (words) vs. tokens (word occurrences)
- Broadly: most word types are rare ones
- Specifically:
  - Rank word types by token frequency
  - Frequency inversely proportional to rank
- Not special to language: randomly generated character strings have this property (try it!)
Parameter Estimation

- Maximum likelihood estimates won’t get us very far

$$\hat{P}(w|w_{-1}) = \frac{c(w_{-1}, w)}{\sum_{w'} c(w_{-1}, w')}$$

- Need to *smooth* these estimates

- General method (procedurally)
  - Take your empirical counts
  - Modify them in various ways to improve estimates

- General method (mathematically)
  - Often can give estimators a formal statistical interpretation
  - … but not always
  - Approaches that are mathematically obvious aren’t always what works
Smoothing

- We often want to make estimates from sparse statistics:

  \[ P(w \mid \text{denied the}) \]
  3 allegations
  2 reports
  1 claims
  1 request
  7 total

- Smoothing flattens spiky distributions so they generalize better

  \[ P(w \mid \text{denied the}) \]
  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! We’ll illustrate with bigrams today (h = previous word, could be anything).
Smoothing: Add-One, Etc.

- Classic solution: add counts (Laplace smoothing / Dirichlet prior)
  
  \[ P_{\text{add-}\delta}(x) = \frac{c(x) + \delta}{\sum_{x'}(c(x') + \delta)} \]

  - Add-one smoothing especially often talked about

- For a bigram distribution, can add counts shaped like the unigram:
  
  \[ P_{\text{dir}}(w|w_{-1}) = \frac{c(w_{-1}, w) + k\hat{P}(w)}{\left( \sum_{w'} c(w_{-1}, w') \right) + k} \]

  - Can consider hierarchical formulations: trigram is recursively centered on smoothed bigram estimate, etc [MacKay and Peto, 94]

- Can be derived from Dirichlet / multinomial conjugacy: prior shape shows up as pseudo-counts

- Problem: works quite poorly!
Linear Interpolation

- Problem: $\hat{P}(w|w_{-1}, w_{-2})$ is supported by few counts
- Classic solution: mixtures of related, denser histories, e.g.:

$$\lambda\hat{P}(w|w_{-1}, w_{-2}) + \lambda'\hat{P}(w|w_{-1}) + \lambda''\hat{P}(w)$$

- The mixture approach tends to work better than the Dirichlet prior approach for several reasons
  - Can flexibly include multiple back-off contexts, not just a chain
  - Often multiple weights, depending on bucketed counts
  - Good ways of learning the mixture weights with EM (later)
  - Not entirely clear why it works so much better

- All the details you could ever want: [Chen and Goodman, 98]
Held-Out Data

- Important tool for optimizing how models generalize:
  - Set a small number of hyperparameters that control the degree of smoothing by maximizing the (log-)likelihood of held-out data
  - Can use any optimization technique (line search or EM usually easiest)

- Examples:

\[
P_{\text{dir}}(w|w_{-1}, k) = \frac{c(w_{-1}, w) + k\hat{P}(w)}{\left(\sum_{w'} c(w_{-1}, w')\right) + k}
\]

\[
P_{\text{lin}}(w|w_{-1}, \lambda, \lambda', \lambda'') = \lambda\hat{P}(w|w_{-1}, w_{-2}) + \lambda'\hat{P}(w|w_{-1}) + \lambda''\hat{P}(w)
\]
Held-Out Reweighting

- What’s wrong with add-d smoothing?
- Let’s look at some real bigram counts [Church and Gale 91]:

<table>
<thead>
<tr>
<th>Count in 22M Words</th>
<th>Actual c* (Next 22M)</th>
<th>Add-one’s c*</th>
<th>Add-0.0000027’s c*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.448</td>
<td>2/7e-10</td>
<td>~1</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>3/7e-10</td>
<td>~2</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>4/7e-10</td>
<td>~3</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>5/7e-10</td>
<td>~4</td>
</tr>
<tr>
<td>5</td>
<td>4.21</td>
<td>6/7e-10</td>
<td>~5</td>
</tr>
</tbody>
</table>

| Mass on New | 9.2% | ~100% | 9.2% |
| Ratio of 2/1 | 2.8  | 1.5   | ~2   |

- Big things to notice:
  - Add-one vastly overestimates the fraction of new bigrams
  - Add-anything vastly underestimates the ratio 2*/1*
- One solution: use held-out data to predict the map of c to c*
Good-Turing Reweighting I

- We’d like to not need held-out data (why?)
- Idea: leave-one-out validation
  - \( N_k \): number of types which occur \( k \) times in the entire corpus
  - Take each of the \( c \) tokens out of corpus in turn
  - \( c \) “training” sets of size \( c-1 \), “held-out” of size \( 1 \)
  - How many “held-out” tokens are unseen in “training”? 
    - \( N_1 \)
  - How many held-out tokens are seen \( k \) times in training? 
    - \((k+1)N_{k+1}\)
  - There are \( N_k \) words with training count \( k \)
  - Each should occur with expected count 
    - \((k+1)N_{k+1}/N_k\)
  - Each should occur with probability: 
    - \((k+1)N_{k+1}/(cN_k)\)
Problem: what about “the”? (say k=4417)
- For small k, $N_k > N_{k+1}$
- For large k, too jumpy, zeros wreck estimates

Simple Good-Turing [Gale and Sampson]: replace empirical $N_k$ with a best-fit power law once count counts get unreliable
Good-Turing Reweighting III

- **Hypothesis:** counts of $k$ should be $k^* = (k+1)N_{k+1}/N_k$

<table>
<thead>
<tr>
<th>Count in 22M Words</th>
<th>Actual c* (Next 22M)</th>
<th>GT’s c*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.448</td>
<td>0.446</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>1.26</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>2.24</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>3.24</td>
</tr>
<tr>
<td>Mass on New</td>
<td>9.2%</td>
<td>9.2%</td>
</tr>
</tbody>
</table>

- **Katz Smoothing**
  - Use GT discounted *bigram* counts (roughly – Katz left large counts alone)
  - Whatever mass is left goes to empirical unigram

$$P_{katz}(w|w') = \frac{c^*(w', w)}{c(w')} + \alpha(w')\hat{P}(w)$$
Kneser-Ney smoothing: very successful estimator using two ideas

Idea 1: observed n-grams occur more in training than they will later:

- Absolute Discounting
  - Save ourselves some time and just subtract 0.75 (or some d)
  - Maybe have a separate value of d for very low counts

<table>
<thead>
<tr>
<th>Count in 22M Words</th>
<th>Avg in Next 22M</th>
<th>Good-Turing c*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.448</td>
<td>0.446</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>1.26</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>2.24</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>3.24</td>
</tr>
</tbody>
</table>

$$P_{ad}(w|w') = \frac{c(w', w) - d}{c(w')} + \alpha(w')\hat{P}(w)$$
Kneser-Ney: Continuation

- **Idea 2: Type-based fertility rather than token counts**
  - Shannon game: There was an unexpected ____?
    - delay?
    - Francisco?
  - “Francisco” is more common than “delay”
  - … but “Francisco” always follows “San”
  - … so it’s less “fertile”

- **Solution: type-continuation probabilities**
  - In the back-off model, we don’t want the probability of w as a unigram
  - Instead, want the probability that w is *allowed in a novel context*
  - For each word, count the number of bigram types it completes

\[
P(w) \propto \sum_{w'} c(w', w)
\]

\[
P_c(w) \propto |w' : c(w', w) > 0|
\]
Kneser-Ney smoothing combines these two ideas
- Absolute discounting
  \[ P(w|w') = \frac{c(w', w) - d}{c(w')} + \alpha(w') P_c(w) \]
- Lower order continuation probabilities
  \[ P_c(w) \propto |w' : c(w', w) > 0| \]

KN smoothing repeatedly proven effective (ASR, MT, …)

[Teh, 2006] shows KN smoothing is a kind of approximate inference in a hierarchical Pitman-Yor process (and better approximations are superior to basic KN)
What Actually Works?

- **Trigrams and beyond:**
  - Unigrams, bigrams generally useless
  - Trigrams much better (when there’s enough data)
  - 4-, 5-grams really useful in MT, but not so much for speech

- **Discounting**
  - Absolute discounting, Good-Turing, held-out estimation, Witten-Bell

- **Context counting**
  - Kneser-Ney construction of lower-order models

- See [Chen+Goodman] reading for tons of graphs!
Data >> Method?

- Having more data is better…

- … but so is using a better estimator
- Another issue: N > 3 has huge costs in speech recognizers
Tons of Data?

[Brants et al, 2007]
Beyond N-Gram LMs

- Lots of ideas we won’t have time to discuss:
  - Caching models: recent words more likely to appear again
  - Trigger models: recent words trigger other words
  - Topic models

- A few recent ideas
  - Syntactic models: use tree models to capture long-distance syntactic effects [Chelba and Jelinek, 98]
  - Discriminative models: set n-gram weights to improve final task accuracy rather than fit training set density [Roark 05, for ASR; Liang et al. 06, for MT]
  - Neural Network language models [Bengio et al. 03, Mikolov et al. 11]
  - Compressed LMs [Pauls & Klein 11, Heafield 11]
What’s Next?

- Finish up noisy-channel models and language modeling

Next Topic: Classification
- Naive Bayes vs. Maximum Entropy vs. Neural Networks
- We introduce a single new global variable
- Still a very simplistic model family
- Lets us model hidden properties of text, but only very non-local ones…
- In particular, we can only model properties which are largely invariant to word order (like topic)

- If you are not fully comfortable with conditional probabilities and maximum likelihood estimators are, read up!

- Reading on the web
- Assignment 1 is already out, due in two weeks!