N-Grams and Corpus Linguistics

Jugal Kalita

(Adapted from Kathy McCoy, University of Delaware)
Sequences of Words

• Up to this point we’ve mostly been discussing words in isolation
• Now we’re switching to sequences of words
• And we’re going to worry about assigning probabilities to sequences of words
Why count word sequences?

• We may want to assign a probability to a sentence or…
• We may want to predict the next word…
• Why?

• Lots of applications
Real-Word Spelling Errors

• Words that sound the same
  – Their/they’re/there
  – To/too/two
  – Weather/whether
  – Peace/piece
  – You’re/your

• Typos that result in real words
  – Lave for Have
Real Word Spelling Errors

• Collect a set of common pairs of confusions
• Whenever a member of this set is encountered, compute the probability of the sentence in which it appears
• Substitute the other possibilities and compute the probability of the resulting sentence
• Choose the higher one
Next Word Prediction

• From a NY Times story...
  – Stocks ...
  – 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
– Stocks plunged this morning, despite a cut in interest rates by the Federal Reserve, as Wall Street began trading for the first time since last …
– Stocks plunged this morning, despite a cut in interest rates by the Federal Reserve, as Wall Street began trading for the first time since last Tuesday's terrorist attacks.
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
Claim

• A useful part of the knowledge needed to allow Word Prediction can be captured using simple statistical techniques

• In particular, we'll rely on the notion of the probability of a sequence (a phrase, a sentence)
Applications

• Why do we want to predict a word, given some preceding words?
  – Rank the likelihood of sequences containing various alternative hypotheses, e.g. for automatic speech recognition
  Theatre owners say popcorn/unicorn sales have doubled...
  – Assess the likelihood/goodness of a sentence, e.g. for text generation or machine translation
  The doctor recommended a cat scan.
  El doctor recomendó una exploración del gato.
N-Gram Models of Language

• 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
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?
Terminology

• **Sentence**: unit of written language
• **Utterance**: unit of spoken language
• **Word Form**: the inflected form that appears in the corpus
• **Lemma**: an abstract form, shared by word forms having the same stem, part of speech, and word sense
• **Types**: number of distinct words in a corpus (vocabulary size)
• **Tokens**: total number of words
Corpora

• Corpora are online collections of text and speech
  – Brown Corpus: 1 million words from 500 texts, divided into genres, collected in 1963-64, need to pay
  – American National Corpus, 22 million words, 15 million are freely available, 1990 onwards
  – Susanne Corpus, 128K subset of the Brown corpus, free
  – Wall Street Journal, 30 million words, need to pay
  – AP news
  – Hansards
  – DARPA/NIST text/speech corpora (Call Home, ATIS, switchboard, Broadcast News, TDT, Communicator)
  – TRAINS, Radio News
Chain Rule

- Recall the definition of conditional probabilities
  \[ P(A \mid B) = \frac{P(A \land B)}{P(B)} \]

- Rewriting
  \[ P(A \land B) = P(A \mid B)P(B) \]

- Or...
  \[ P(The \ big) = P(big \mid the)P(the) \]
Example

- The big red dog

- $P(\text{The}) \cdot P(\text{big}|\text{the}) \cdot P(\text{red}|\text{the big}) \cdot P(\text{dog}|\text{the big red})$

- Better $P(\text{The}|<\text{Beginning of sentence}>)$ written as $P(\text{The}|<S>)$
General Case

- The word sequence from position 1 to $n$ is $\mathcal{W}^n_1$
- So the probability of a sequence is

$$P(\mathcal{W}^n_1) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1})$$

$$= P(w_1) \prod_{k=2}^{n} P(w_k \mid w_1^{k-1})$$
Unfortunately

- That doesn’t help since it’s unlikely we’ll ever gather the right statistics for the prefixes.
Markov Assumption

• Assume that the entire prefix history isn’t necessary.
• In other words, an event doesn’t depend on all of its history, just a fixed length near history
Markov Assumption

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

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

The big red dog

• Unigrams: $P(\text{dog})$
• Bigrams: $P(\text{dog}|\text{red})$
• Trigrams: $P(\text{dog}|\text{big red})$
• Four-grams: $P(\text{dog}|\text{the big red})$

In general, we’ll be dealing with

$P(\text{Word} | \text{Some fixed prefix})$
Caveat

• The formulation \( P(\text{Word}| \text{Some fixed prefix}) \) is not really appropriate in many applications.
• It is if we’re dealing with real time speech where we only have access to prefixes.
• But if we’re dealing with text we already have the right and left contexts. There’s no a priori reason to stick to left contexts.
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 test set
  – cross validation
  – results tested for statistical significance
A Simple Example

- \( P(\text{I want to eat Chinese food}) = P(\text{I | <start>}) \ P(\text{want | I}) \ P(\text{to | want}) \ P(\text{eat | to}) \ P(\text{Chinese | eat}) \ P(\text{food | Chinese}) \)
A Bigram Grammar Fragment from BERP (Berkeley Restaurant Corpus, 7500 sentences with 500 words, 6.5 hours of speech)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>eat on</td>
<td>.16</td>
<td>eat Thai</td>
<td>.03</td>
</tr>
<tr>
<td>eat some</td>
<td>.06</td>
<td>eat breakfast</td>
<td>.03</td>
</tr>
<tr>
<td>eat lunch</td>
<td>.06</td>
<td>eat in</td>
<td>.02</td>
</tr>
<tr>
<td>eat dinner</td>
<td>.05</td>
<td>eat Chinese</td>
<td>.02</td>
</tr>
<tr>
<td>eat at</td>
<td>.04</td>
<td>eat Mexican</td>
<td>.02</td>
</tr>
<tr>
<td>eat a</td>
<td>.04</td>
<td>eat tomorrow</td>
<td>.01</td>
</tr>
<tr>
<td>eat Indian</td>
<td>.04</td>
<td>eat dessert</td>
<td>.007</td>
</tr>
<tr>
<td>eat today</td>
<td>.03</td>
<td>eat British</td>
<td>.001</td>
</tr>
<tr>
<td></td>
<td>Probability</td>
<td></td>
<td>Probability</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
<td>----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>&lt;start&gt; I</td>
<td>.25</td>
<td>want some</td>
<td>.04</td>
</tr>
<tr>
<td>&lt;start&gt; I’d</td>
<td>.06</td>
<td>want Thai</td>
<td>.01</td>
</tr>
<tr>
<td>&lt;start&gt; Tell</td>
<td>.04</td>
<td>to eat</td>
<td>.26</td>
</tr>
<tr>
<td>&lt;start&gt; I’m</td>
<td>.02</td>
<td>to have</td>
<td>.14</td>
</tr>
<tr>
<td>I want</td>
<td>.32</td>
<td>to spend</td>
<td>.09</td>
</tr>
<tr>
<td>I would</td>
<td>.29</td>
<td>to be</td>
<td>.02</td>
</tr>
<tr>
<td>I don’t</td>
<td>.08</td>
<td>British food</td>
<td>.60</td>
</tr>
<tr>
<td>I have</td>
<td>.04</td>
<td>British restaurant</td>
<td>.15</td>
</tr>
<tr>
<td>want to</td>
<td>.65</td>
<td>British cuisine</td>
<td>.01</td>
</tr>
<tr>
<td>want a</td>
<td>.05</td>
<td>British lunch</td>
<td>.01</td>
</tr>
</tbody>
</table>
• \( P(\text{I want to eat British food}) = P(\text{I}|<\text{start}>) \ P(\text{want}|\text{I}) \ P(\text{to}|\text{want}) \ P(\text{eat}|\text{to}) \ P(\text{British}|\text{eat}) \ P(\text{food}|\text{British}) = 0.25 \times 0.32 \times 0.65 \times 0.26 \times 0.001 \times 0.60 = 0.000080 \)

• vs. \( I \text{ want to eat Chinese food} = 0.00015 \)

• Probabilities seem to 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
An Aside on Logs

• You don’t really do all those multiplies. The numbers are too small and lead to underflows.

• Convert the probabilities to logs and then do additions.

• To get the real probability (if you need it) go back to the antilog.
How do we get the N-gram probabilities?

• N-gram models can be trained by counting and normalization
### BERP Bigram Counts

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>8</td>
<td>1087</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>want</td>
<td>3</td>
<td>0</td>
<td>786</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>860</td>
<td>3</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>19</td>
<td>2</td>
<td>52</td>
</tr>
<tr>
<td>Chinese</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>food</td>
<td>19</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
BERP Bigram Probabilities

- Normalization: divide each row's counts by appropriate unigram counts for $w_{n-1}$

<table>
<thead>
<tr>
<th></th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>3437</td>
<td>1215</td>
<td>3256</td>
<td>938</td>
<td>213</td>
<td>1506</td>
</tr>
</tbody>
</table>

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

- Maximum Likelihood Estimation (MLE): relative frequency of e.g.

\[
\frac{f \text{rel}(w_1, w_2)}{f \text{rel}(w_1)}
\]
## BERP Table: Bigram Probabilities

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>.0023</td>
<td>.32</td>
<td>0</td>
<td>.0038</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>want</td>
<td>.0025</td>
<td>0</td>
<td>.65</td>
<td>0</td>
<td>.0049</td>
<td>.0066</td>
<td>.0049</td>
</tr>
<tr>
<td>to</td>
<td>.00092</td>
<td>0</td>
<td>.0031</td>
<td>.26</td>
<td>.00092</td>
<td>0</td>
<td>.0037</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>.0021</td>
<td>0</td>
<td>.020</td>
<td>.0021</td>
<td>.055</td>
</tr>
<tr>
<td>Chinese</td>
<td>.0094</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.56</td>
<td>.0047</td>
</tr>
<tr>
<td>food</td>
<td>.013</td>
<td>0</td>
<td>.011</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>.0087</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.0022</td>
<td>0</td>
</tr>
</tbody>
</table>
What do we learn about the language?

• What's being captured with ...
  – $P(\text{want} \mid I) = .32$
  – $P(\text{to} \mid \text{want}) = .65$
  – $P(\text{eat} \mid \text{to}) = .26$
  – $P(\text{food} \mid \text{Chinese}) = .56$
  – $P(\text{lunch} \mid \text{eat}) = .055$

• What about...
  – $P(I \mid I) = .0023$
  – $P(I \mid \text{want}) = .0025$
  – $P(I \mid \text{food}) = .013$
- $P(I \mid I) = .0023$ I want
- $P(I \mid \text{want}) = .0025$ I want I want
- $P(I \mid \text{food}) = .013$ the kind of food I want is ...
Generation – just a test

• Choose N-Grams according to their probabilities and string them together
• For bigrams – start by generating a word that has a high probability of starting a sentence, then choose a bigram that is high given the first word selected, and so on.
• See we get better with higher-order n-grams
Approximating Shakespeare

• As we increase the value of N, the accuracy of the 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 about a 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 worse: 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
Some Useful Observations

• A small number of events occur with high frequency
  – You can collect reliable statistics on these events with relatively small samples

• A large number of events occur with small frequency
  – You might have to wait a long time to gather statistics on the low frequency events

• Because any corpus is limited, some perfectly acceptable word sequences are bound to be missing from it.

• Thus, there is bound to be a very large number of 0 entries in any N-gram matrix.
Some Useful Observations

• Some zeroes are really zeroes
  – Meaning that they represent events that can’t or shouldn’t occur

• On the other hand, some zeroes aren’t really zeroes
  – They represent low frequency events that simply didn’t occur in the corpus

• Because of these reasons, we want to modify the probability (frequency) computation, focusing on those N-grams for which we obtain 0 probability (frequency)
Smoothing

• Every n-gram training matrix is sparse, even for very large corpora (Zipf’s law)
• Solution: estimate the likelihood of unseen n-grams
• Problems: how do you adjust the rest of the corpus to accommodate these ‘phantom’ n-grams?
• This is called **Smoothing**: We shave a little bit of probability mass from the higher counts and assign it on the zero counts
Problem

• Let’s assume we’re using N-grams
• How can we assign a probability to a sequence where one of the component n-grams has a value of zero
• Assume all the words are known and have been seen
• Here are some ideas for improving N-gram counts (probabilities)
  – Go to a lower order n-gram
  – Back off from bigrams to unigrams
  – Replace the zero with something else
Add-One (Laplace)

- Make the zero counts 1.
- Rationale: They’re just events you haven’t seen yet. If you had seen them, chances are you would only have seen them once... so make the count equal to 1.
Add-one Smoothing

• For unigrams:
  – Add 1 to every word (type) count
  – Normalize by $N$ (tokens) /$(N$ (tokens) +$V$ (types))
  – Smoothed count (adjusted for additions to $N$) is
    $$\frac{(c_i+1)N}{N+V}$$
  – Normalize by $N$ to get the new unigram probability:

• For bigrams:
  – Add 1 to every bigram $c(w_{n-1} w_n)$ + 1
  – Incr unigram count by vocabulary size $c(w_{n-1}) + V$
### Original BERP Counts

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>8</td>
<td>1087</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>want</td>
<td>3</td>
<td>0</td>
<td>786</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>860</td>
<td>3</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>19</td>
<td>2</td>
<td>52</td>
</tr>
<tr>
<td>Chinese</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>food</td>
<td>19</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
# BERP Table: Bigram Probabilities

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>.0023</td>
<td>.32</td>
<td>0</td>
<td>.0038</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>want</td>
<td>.0025</td>
<td>0</td>
<td>.65</td>
<td>0</td>
<td>.0049</td>
<td>.0066</td>
<td>.0049</td>
</tr>
<tr>
<td>to</td>
<td>.00092</td>
<td>0</td>
<td>0</td>
<td>.0031</td>
<td>.26</td>
<td>.00092</td>
<td>0</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>.0021</td>
<td>0</td>
<td>.020</td>
<td>.0021</td>
<td>.055</td>
</tr>
<tr>
<td>Chinese</td>
<td>.0094</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.56</td>
<td>.0047</td>
</tr>
<tr>
<td>food</td>
<td>.013</td>
<td>0</td>
<td>.011</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>.0087</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.0022</td>
<td>0</td>
</tr>
</tbody>
</table>
BERP After Add-One

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>.0018</td>
<td>.22</td>
<td>.00020</td>
<td>.0028</td>
<td>.00020</td>
<td>.00020</td>
<td>.00020</td>
</tr>
<tr>
<td>want</td>
<td>.0014</td>
<td>.00035</td>
<td><strong>.28</strong></td>
<td>.00035</td>
<td>.0025</td>
<td>.0032</td>
<td>.0025</td>
</tr>
<tr>
<td>to</td>
<td>.00082</td>
<td>.00021</td>
<td>.0023</td>
<td>.18</td>
<td>.00082</td>
<td>.00021</td>
<td>.0027</td>
</tr>
<tr>
<td>eat</td>
<td>.00039</td>
<td>.00039</td>
<td>.0012</td>
<td>.00039</td>
<td>.0078</td>
<td>.0012</td>
<td>.021</td>
</tr>
<tr>
<td>Chinese</td>
<td>.0016</td>
<td>.00055</td>
<td>.00055</td>
<td>.00055</td>
<td>.0055</td>
<td>.066</td>
<td>.011</td>
</tr>
<tr>
<td>food</td>
<td>.0064</td>
<td>.00032</td>
<td>.0058</td>
<td>.00032</td>
<td>.00032</td>
<td>.00032</td>
<td>.00032</td>
</tr>
<tr>
<td>lunch</td>
<td>.0024</td>
<td>.00048</td>
<td>.00048</td>
<td>.00048</td>
<td>.00048</td>
<td>.00096</td>
<td>.00048</td>
</tr>
</tbody>
</table>

Was .65
## Add-One Smoothed BERP Reconstituted

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>6</td>
<td>740</td>
<td>.68</td>
<td>10</td>
<td>.68</td>
<td>.68</td>
<td>.68</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>.42</td>
<td>331</td>
<td>.42</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>.69</td>
<td>8</td>
<td>594</td>
<td>3</td>
<td>.69</td>
<td>9</td>
</tr>
<tr>
<td>eat</td>
<td>.37</td>
<td>.37</td>
<td>1</td>
<td>.37</td>
<td>7.4</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Chinese</td>
<td>.36</td>
<td>.12</td>
<td>.12</td>
<td>.12</td>
<td>.12</td>
<td>15</td>
<td>.24</td>
</tr>
<tr>
<td>food</td>
<td>10</td>
<td>.48</td>
<td>9</td>
<td>.48</td>
<td>.48</td>
<td>.48</td>
<td>.48</td>
</tr>
<tr>
<td>lunch</td>
<td>1.1</td>
<td>.22</td>
<td>.22</td>
<td>.22</td>
<td>.22</td>
<td>.44</td>
<td>.22</td>
</tr>
</tbody>
</table>
Problems with Laplace Smoothing

- Discount: ratio of new counts to old (e.g. add-one smoothing changes the BERP count (to|want) from 786 to 331 ($d_c = .42$) and $p(to|want)$ from .65 to .28)

- Problem: add one smoothing changes counts drastically:
  - too much weight given to unseen ngrams
  - in practice, unsmoothed bigrams often work better!
A zero N-gram is just an N-gram you haven’t seen yet…but every N-gram in the corpus was unseen once…so...

- How many times did we see an N-gram for the first time?
  Once for each N-gram type (T): We have to categorize the N-grams into some types
- Est. total probability mass of unseen bigrams as
  \[
  \frac{T}{N+T}
  \]
- View training corpus as series of events, one for each token (N) and one for each new type (T)
Witten-Bell

• Think about the occurrence of an unseen item (word, bigram, etc) as an event.
• The probability of such an event can be measured in a corpus by just looking at how often it happens.
• Just take the single word case first.
• Assume a corpus of $N$ tokens and $T$ types.
• How many times was an as yet unseen type encountered?
Witten Bell

• First compute the probability of an unseen event occurring
• Then distribute that probability mass among the as yet unseen types (the ones with zero counts)
Probability of an Unseen Event

• Simple case of unigrams
• $T$ is the number of events that are seen for the first time in the corpus
• This is just the number of types since each type had to occur for a first time once
• Types can be POS or something like that. They can be just the words.
• $N$ is just the number of observations; i.e., $N$ is the number of unigrams here.

$$\frac{T}{N + T}$$
Distributing Evenly

- The amount of probability mass to be distributed is

\[ \frac{T}{N + T} \]

- The number of events with count zero, i.e., the number of unseen N-grams

- So distributing evenly over all N-grams gets us

\[ \frac{1}{Z} \frac{T}{N + T} \]
Caveat

• The unigram case is weird…
  – Z is the number of things with count zero
  – Ok, so that’s the number of things we didn’t see at all. Huh?
• Fortunately it makes more sense in the N-gram case.
  – Take Shakespeare… Recall that he produced only 29,000 types. So there are potentially $29,000^2$ bigrams. Of which only 300k occur, so Z is $29,000^2 - 300k$
In the case of bigrams, not all conditioning events are equally promiscuous
- $P(x|\text{the})$ vs $P(x|\text{going})$

So distribute the mass assigned to the zero count bigrams according to their promiscuity
- This means condition the redistribution on how many different types occurred with a given prefix
Distributing Among the Zeros

• If a bigram “wx wi” has a zero count

\[ P(w_i \mid w_x) = \frac{1}{Z(w_x)} \frac{T(w_x)}{N(w_x) + T(w_x)} \]

- Number of bigrams starting with wx that were not seen
- Number of bigram types starting with wx
- Actual frequency of bigrams beginning with wx
## Original BERP Counts

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>8</td>
<td>1087</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>want</td>
<td>3</td>
<td>0</td>
<td>786</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>860</td>
<td>3</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>19</td>
<td>2</td>
<td>52</td>
</tr>
<tr>
<td>Chinese</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>food</td>
<td>19</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Witten-Bell Smoothed and Reconstituted Counts

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>8</td>
<td>1060</td>
<td>.062</td>
<td>13</td>
<td>.062</td>
<td>.062</td>
<td>.062</td>
</tr>
<tr>
<td>want</td>
<td>3</td>
<td>.046</td>
<td>740</td>
<td>.046</td>
<td>6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>.085</td>
<td>10</td>
<td>827</td>
<td>3</td>
<td>.085</td>
<td>12</td>
</tr>
<tr>
<td>eat</td>
<td>.075</td>
<td>.075</td>
<td>2</td>
<td>.075</td>
<td>17</td>
<td>2</td>
<td>46</td>
</tr>
<tr>
<td>Chinese</td>
<td>2</td>
<td>.012</td>
<td>.012</td>
<td>.012</td>
<td>.012</td>
<td>109</td>
<td>1</td>
</tr>
<tr>
<td>food</td>
<td>18</td>
<td>.059</td>
<td>16</td>
<td>.059</td>
<td>.059</td>
<td>.059</td>
<td>.059</td>
</tr>
<tr>
<td>lunch</td>
<td>4</td>
<td>.026</td>
<td>.026</td>
<td>.026</td>
<td>.026</td>
<td>1</td>
<td>.026</td>
</tr>
</tbody>
</table>
Good-Turing Discounting

• Re-estimate amount of probability mass for zero (or low count) ngrams by looking at ngrams with higher counts
  – Estimate
    \[ c^* = (c + 1)^{\frac{Nc+1}{Nc}} \]
  – E.g. \( N_0 \)’s adjusted count is a function of the count of ngrams that occur once, \( N_1 \)
  – Assumes:
    • word bigrams follow a binomial distribution
    • We know number of unseen bigrams (VxV-seen)
Backoff methods (e.g. Katz ‘87)

• For e.g. a trigram model
  – Compute unigram, bigram and trigram probabilities
  – In use:
    • Where trigram unavailable back off to bigram if available, o.w. unigram probability
    • E.g An omnivorous *unicorn*
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 a corpus